Random Forest Analogues for Mixture Discriminant Analysis

RANDOM FOREST ANALOGUES FOR MIXTURE DISCRIMINANT ANALYSIS

BY

MUZ MALLO, M.Sc.

A THESIS

SUBMITTED TO THE DEPARTMENT OF MATHEMATICS & STATISTICS

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

Doctor of Philosophy (2019) McMaster University

(Mathematics & Statistics) Hamilton, Ontario, Canada

TITLE:	Random Forest Analogues for Mixture Discriminant Analysis
AUTHOR:	Muz Mallo M.Sc. (Statistics), University of Toronto, Toronto, Canada
SUPERVISOR:	Dr. Paul D. McNicholas
NUMBER OF PAGES:	xv, 123

For we are His workmanship, created in Christ Jesus unto good works, which God hath before ordained that we should walk in them.

(Ephesians 2:10)

For of Him, and through Him, and to Him, are all things: to whom be glory for ever. Amen.

(Romans 11:36)

Abstract

Finite mixture modelling is a powerful and well-developed paradigm, having proven useful in unsupervised learning and, to a lesser extent supervised learning (mixture discriminant analysis), especially in the case(s) of data with local variation and/or latent variables. It is the aim of this thesis to improve upon mixture discriminant analysis by introducing two types of random forest analogues which are called MixForests. The first MixForest is based on Gaussian mixture models from the famous family of Gaussian parsimonious clustering models and will be useful in classifying lower dimensional data. The second MixForest extends the technique to higher dimensional data via the use of mixtures of factor analyzers from the well-known family of parsimonious Gaussian mixture models. MixForests will be utilized in the analysis of real data to demonstrate potential increases in classification accuracy as well as inferential procedures such as generalization error estimation and variable importance measures.

Acknowledgements

Firstly, I would like to thank Dr. Paul D. McNicholas. Paul's guidance and support are far above what I ever expected. I would be hard-pressed to find a better supervisor.

I extend my gratitude towards Dr. Bartosz Protas and Dr. Fei Chiang for serving on my committee. I would like to thank Dr. Alison Weir and Dr. Jerry Brunner without whose support I would not have gone the distance.

Thanks, Mom and Dad (Arif and Reba Mallo), for everything. I'd also like to thank my family members, especially Uncle George, Aunt Bubbly and my grandparents.

I'd like to thank my friends Peter Olsen, Kerri Fuller, and Rudolf Dosen for their emotional support.

I'd like to thank my friends at the University of Toronto, especially those in the Department of Statistics, who made my graduate years there so educational and enjoyable. Also, I'd like to thank my compatriots in the McNicholas research group, who value coffee as much as I do.

Thank you, Christine Hollingshead, for all your prayers and support.

It would be difficult not to mention Timmy the cat and Joey the cat, who help keep things in perspective.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Discriminant Analysis

The primary objective of discriminant analysis (DA) is the development of a rule or classifier whereby a newly considered entity can be allocated to a previously observed subpopulation. Over the many decades of statistical theory, the conditions and considerations of DA have varied depending on the context. In this setting, DA will be defined using the following conditions.

The set of observations will be comprised of N p-dimensional vectors of measurements, denoted $\mathbf{x}_1, \ldots, \mathbf{x}_N$. Of the N observations, n will be known *a priori* to come from one of G observed classes, while $N-n$ will be unlabelled. DA is based upon only the n labelled observations and will be used to infer the membership of the remaining observations. In many ways the evolution of DA mirrors the progress of statistical inference as a whole. One author even suggested that the entire topic of multivariate

analysis can be considered an extension of the discriminant function. DA arose from the science of biological taxonomy and is widely applicable to many diverse fields including physics, social science, engineering, medicine, pattern recognition, etc. DA is at the core of the science of statistics.

1.2 Mixture Discriminant Analysis

Mixture discriminant analysis (MDA) is the use of finite mixture models (FMMs) in DA. While the term MDA is a relatively recent invention, finite mixtures have been employed in DA as long ago as the late 1800s by Newcomb (1886) and by Pearson (1894). The p-dimensional random vector $\mathbf{X} = (X_1, \ldots, X_p)'$ is said to arise from a mixture distribution if its density can be written

$$f(\mathbf{x}|\boldsymbol{\vartheta}) = \sum_{g=1}^{G} \pi_g f_g(\mathbf{x}|\boldsymbol{\theta}_g),$$

where $\pi_g > 0$ represents the gth mixing proportion, $\sum_{g=1}^{G} \pi_g = 1$, $\boldsymbol{\vartheta} = (\boldsymbol{\pi}, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_G)$ is the vector of parameters with $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_G)$, and $f_g(\mathbf{x}|\boldsymbol{\theta}_g)$ is the gth component density, and $f(\mathbf{x}|\boldsymbol{\vartheta})$ is then called the G-component finite mixture density. Parametrically speaking, MDA is natural since FMMs (1.2) suppose the existence of subpopulations within an overall population. Although component densities are assumed to be the same type, they can be chosen to suit the need of a particular application (i.e., outlier detection).

With the tremendous work done in the area of finite mixtures, it is a well thought out and flexible modelling paradigm. Parameter estimation has been laid out for

mixtures of various types including multivariate Gaussians (Celeux and Govaert, 1995), multivariate t-factor analyzers (McLachlan *et al.*, 2007), skew-t factor analyzers (Murray *et al.*, 2014), hidden truncation hyperbolic distributions (Murray *et al.*, 2017), generalized hyperbolic distributions (Browne and McNicholas, 2015), multivariate Poisson-log normal (Silva *et al.*, 2017), gamma (Webb, 2000), variance-gamma factor analyzers (McNicholas *et al.*, 2017), etc. For the purposes of this thesis, two families of models will be employed: the family of Gaussian parsimonious clustering models (GPCM; Celeux and Govaert, 1995) and the family of parsimonious Gaussian mixture models (PGMM; McNicholas and Murphy, 2010). Finite mixture models have been applied to diverse fields and studies.

1.3 Random Forests

The main idea of random forests is to generate an ensemble of tree-type classifiers using a modification of bootstrap aggregating (bagging) which minimizes correlation between trees (Breiman, 2001). For DA, significant improvements in classification accuracy can be made by allowing the trees to take a committee vote to place a newly considered entity into a previously observed class (Breiman, 2001).

This method addresses the bias-variance dilemma present in model prediction. The two sources of prediction error are bias and variance. The dilemma is that decreasing one tends to increase the other. Bagging works very well at decreasing prediction error for classifiers which can be characterized as having high variance

and low bias, such as classification trees (Friedman *et al.*, 2001). By taking a consensus amongst the trees the variance of their individual predictions is reduced, thus improving prediction error (Breiman, 1996b).

Random forests also possess other inferential advantages including speedy performance, being able to handle thousands of input variables, variable importance scores, internal unbiased estimates of generalization error, proximity measures between observations, etc. In addition to this, random forests have only one parameter which needs tuning since the trees are not pruned. Thus, they are simple to employ automatically (Breiman and Cutler, 2004). Random forests are considered to be one of the most accurate general-purpose methods available (Biau, 2012).

1.4 Thesis Outline

1.4.1 Chapter 2

Chapter 2 begins with a definition of classification, setting the tone for the rest of the thesis. Next, background information pertaining to classification trees, bagging, bootstrapping, random forests, and inference using random forests is provided.

1.4.2 Chapter 3

Chapter 3 covers material pertaining to Gaussian mixture models, the family of Gaussian parsimonious clustering models, parameter estimation using the expectation maximization algorithm, model-based clustering, mixture discriminant analysis,

model selection, and performance assessment. The mixForest technique is introduced, as well as inference using mixForests. The mixForest is illustrated in the analysis of three datasets: the Pima Indians diabetes data, the wireless indoor localization data, and the Italian wine data. This chapter concludes with a discussion and summary of the results therein.

1.4.3 Chapter 4

Chapter 4 covers the material necessary to create a mixForest for high dimensional data: the factor analysis model, mixtures of factor analyzers, the family of parsimonious Gaussian mixture models, and parameter estimation using the alternating expectation-conditional maximization algorithm. The new mixForest is introduced and utilized in the analysis of the following datasets: the forest types mapping data, the ionosphere data, and the pen-based handwritten digit recognition data. The chapter concludes with a brief discussion of the results therein.

1.4.4 Chapter 5

Advantages, disadvantages, overall performance and other issues implementing the mixForest technique are discussed. Also, future work regarding further development of mixForests will be suggested.

1.5 Impact

The goal of this thesis is to make MDA better. To do this an ensemble of finite mixture models will be constructed based on the random forests method. This will produce a classifier called a MixForest which will yield increases in accuracy, variable importance scores, estimates of generalization error, and the ability to deal with missing data. Additionally, one version of the MixForest will also have the benefit of dimensionality reduction, due to the use of mixtures of factor analyzers.

Chapter 2

Background

2.1 Classification

Because the focus of this paper is improving MDA, we restrict our attention to the process of classification, where the aim is predicting a class label for a previously unlabelled observation. In other words, the job of a classification method is to take a previously unlabelled observation $\mathbf{x}_i$, which consists of p features or measurements, and apply one of G previously observed labels to it. The data is denoted $\mathbf{T} = \{(y_i, \mathbf{x}_i) | i = 1, \ldots, N\}$, where y_i is a class label and $\mathbf{x}_i$ is a p-dimensional feature vector. For each observation, y_i may or may not be known. When none of the y_i are known, then an unsupervised classification (or clustering) technique is needed. If $n < N$ labels are known then either supervised or semi-supervised classification can be used. In semi-supervised classification, the $N - n$ unlabelled observations are labelled based on a model developed using all N observations. However, in

supervised classification, the $N - n$ unlabelled observations are labelled based on a model developed using only the n labelled observations. See McNicholas (2016) for a detailed discussion of clustering, semi-supervised classification and supervised classification using mixture models.

2.2 Classification Trees

According to Hssina *et al.* (2014), there are two notable lineages leading to modern decision trees. From a machine learning perspective, the most popular line of tree-type classifiers began with the work of Quinlan (1986) which introduced the ID3 (Iterative Dichotomiser 3) algorithm. The ID3 algorithm would lead to the C4.5 algorithm (Quinlan, 1993) and the C5.0 algorithm (Quinlan, 1997). Quinlan relates his work to that of Hunt (1962). Alternatively, statisticians widely regard tree-type classifiers to have originated with the work of Morgan and Sonquist (1963), which introduced the automatic interaction detector (AID) algorithm. This evolved into the works of Morgan and Messenger (1973) which introduced the theta AID (THAID) and the work of Kass (1980) which introduced the chi-squared AID (CHAID). Eventually these algorithms gave rise to classification and regression trees (CART; Breiman *et al.*, 1984).

For the purposes of this paper, the scope will be limited to classification trees using the CART methodology. The observations $\mathbf{T} = \{(y_i, \mathbf{x}_i)|i = 1, \ldots, N\}$ consist of p-dimensional feature vectors, with the target/class variable y_i taking values in $\{1, \ldots, G\}$.

The premise behind a classification tree is to perform recursive binary splitting of the feature space according to a measure of goodness until a stopping criterion is met (Breiman *et al.*, 1984). As a result, the feature space is split into rectangular regions/partitions and a simple prediction model (e.g., constant) is used in each region (Friedman *et al.*, 2001). For the purposes of this section the `R` implementation of the CART algorithm in package `rpart` (Therneau *et al.*, 2018) was used to create the trees and the `R` package `rpart.plot` (Milborrow and Milborrow, 2018) for the illustrations.

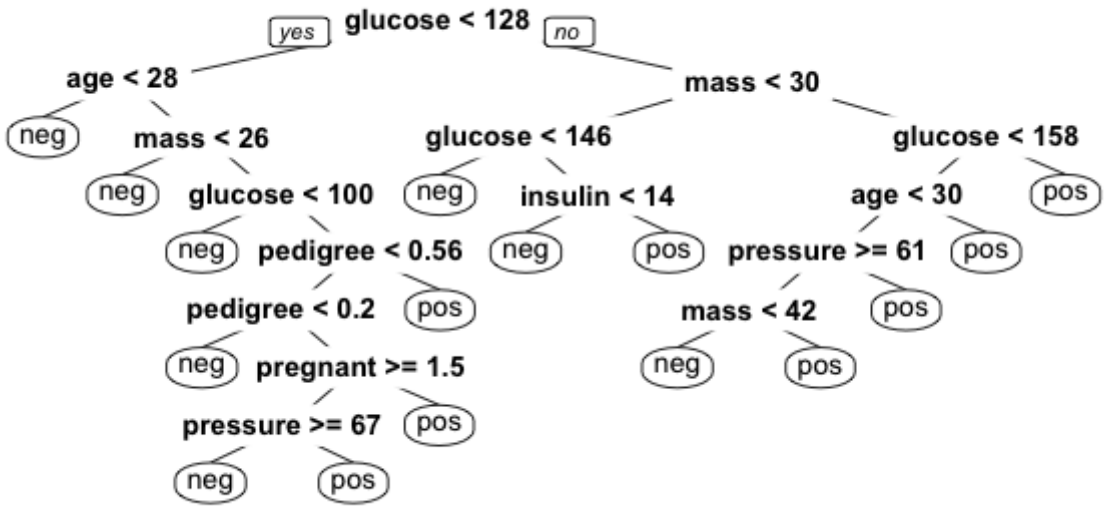

Figure 2.1: An unpruned classification tree grown from the Pima Indians diabetes data.

The root node at the very top of the tree diagram (Figure 2.1), representing the entire sample, is repeatedly split into two sub-nodes. This is done by considering

functions of the proportion of class g in node m:

$$\hat{p}_{mg} = \frac{1}{N_m} \sum_{\mathbf{x}_i \in R_m} I\left(y_i = g\right),$$

where $N_m = \#\left\{\mathbf{x}_i \in R_m\right\}$ and R_m is the region represented by node m. Typically, when dealing with a multi-class problem the Gini index is utilized (Breiman *et al.*, 1984).

The Gini index is a natural choice as a measure of statistical dispersion and can be used to determine node purity. Node purity is the greatest when the node contains only one class, and least when all classes are equally mixed (Breiman *et al.*, 1984). The Gini index is given by

$$\sum_{g \neq g'} \hat{p}_{mg} \hat{p}_{mg'} = \sum_{g=1}^{G} \hat{p}_{mg} \left(1 - \hat{p}_{mg}\right).$$

According to Breiman *et al.* (1984), for each split, all possible features and all possible splits are considered, and the one resulting in the greatest node purity according to the Gini index is chosen. This splitting is done until no further improvements in node purity can be made and the tree will terminate in a set of terminal nodes or leaves. At this point, all observations in node m are classified according to the rule

$$g\left(m\right) = \underset{g}{\operatorname{argmax}}\, \hat{p}_{mg},$$

which represents the majority class in that region (Friedman *et al.*, 2001).

Growing a tree this way often results in a tree with a higher true misclassification

rate (Breiman *et al.*, 1984). Thus, trees are usually pruned by removing sub-nodes of the tree which offer lower classification power. This pruning will result in a reduction in complexity and overfitting, resulting in greater predictive power (Breiman *et al.*, 1984). Figure 2.2 illustrates pruning. In addition to the other benefits, there is a clear improvement in the interpretability of the tree.

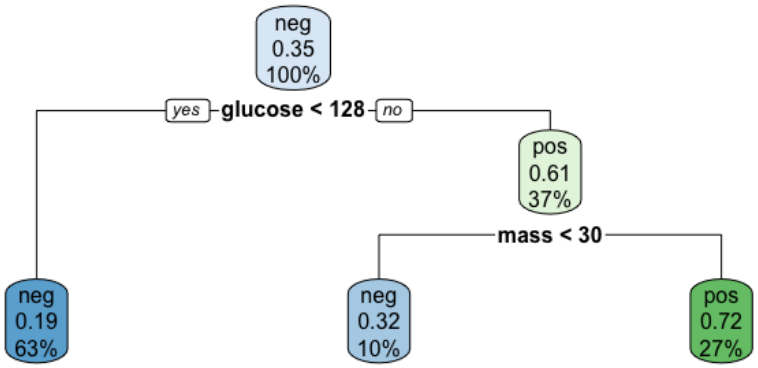

Figure 2.2: A pruned classification tree, grown from the Pima Indians diabetes data.

Classification trees offer many advantages: variable importance can be inferred based on splits, interactions between variables are evident, natural visualization, and easy interpretation to give clear decision rules (Breiman *et al.*, 1984). However, there are drawbacks to using trees. A small change in the training data will lead to a different set of splits, making tree structures unstable (Breiman *et al.*, 1984). This high variance behaviour can be tempered by using the techniques described in Section 2.3 (Friedman *et al.*, 2001).

2.3 Random Forests

2.3.1 Overview

A random forest is a form of supervised learning that generates an ensemble of tree type classifiers using a modification of the bagging method. Also, by adding some randomness, the tree classifiers are grown to be independent of one another yielding better results. The main idea behind random forests and bagging is that the trees can be grown to be independent of one another while maintaining their strength as classifiers, thus producing an aggregated classifier which will yield a performance increase (Breiman, 2001).

2.3.2 Bagging

Bagging is a method for creating an ensemble of learners using the same base learning algorithm, by training each learner on a bootstrap replicate of the original data, and aggregating the ensemble (Breiman, 1996a). There are two steps in bagging: (a) bootstrapping and (b) aggregation. Each time a bootstrap replicate is created, a number of observations are left out of the sample and these are called out-of-bag (OOB) observations (Breiman, 2001). Bagging generally gives us increased accuracy over decision trees as well as the natural ability to obtain an ongoing estimate of the classification error via the OOB error.

Random feature selection at each split can further increase the accuracy of the ensemble, by allowing us to strategically inject randomness (Breiman, 2001). The result, which is the aggregation of more diverse trees, is known as random forests.

Further benefits of random forests include robustness to outliers and noise, speed, internal estimates of error, strength, correlation, variable importance, and simplicity (Breiman, 2001). In the following sections, the construction of random forests is broken into (a) bootstrapping, (b) training the learners with random feature selection, and (c) aggregation.

2.3.3 Bootstrapping

The original set of observed data is denoted $\mathbf{T} = \{(y_i, \mathbf{x}_i)|i = 1, \ldots, N\}$, with $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,p})'$. K bootstrap replicates denoted $\mathbf{T}_{B,1}, \ldots, \mathbf{T}_{B,K}$ are each generated by making $d = N$ uniformly random draws with replacement on $\mathbf{T}$. The ensemble is constructed so that there is a one-to-one correspondence between the bootstrap replicates ($\mathbf{T}_{B,k}$) and the learners. Each time a bootstrap replicate is generated, roughly 37% of the original data points are left out (Breiman, 1996b). To understand this, consider the probability that a data point is left out of a bootstrap replicate, assuming that there are N observations with $d = N$ random draws with replacement:

$$P(\text{leave out an observation}) = \left(\frac{N-1}{N}\right)^d.$$

Looking at Figure 2.3, we see that this number approaches 37%. The intuition from Figure 2.3 makes sense because

$$\lim_{N\to\infty}\left(\frac{N-1}{N}\right)^N = \frac{1}{e} \approx 37\%$$

The fact that we expect around 37% of observations to be omitted from each

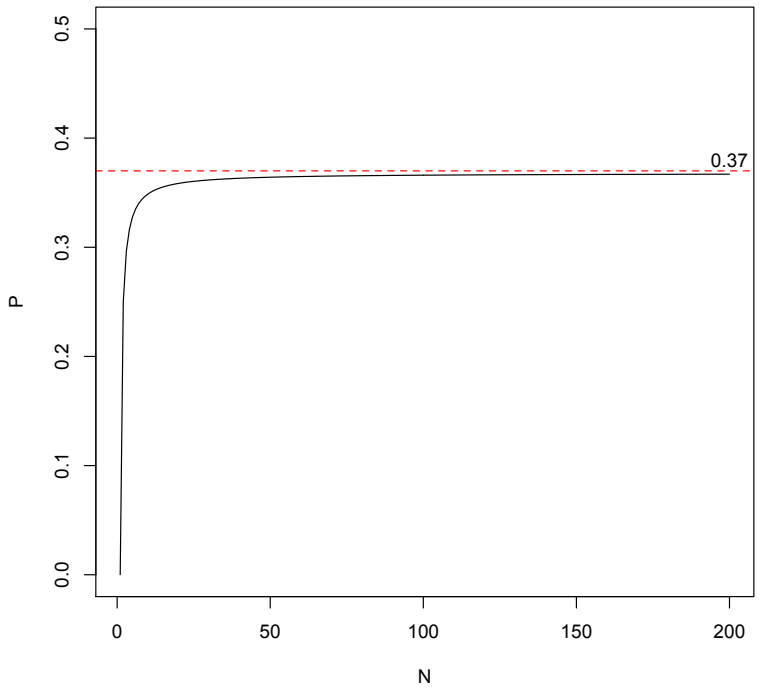

Figure 2.3: Plot of $P = [(N-1)/N]^N$ against N.

bootstrap sample is, perhaps counter-intuitively, an advantage in a number of ways. Besides the usual advantages of the bootstrap, these OOB data points provide a built-in "pseudo test set" because we have the known labels under our MDA scheme. Also, these OOB data points will be the basis of one of our primary inferential quantities.

2.3.4 Random Feature Selection

As described in Section 2.2, when growing a tree all possible splits of all predictors are considered during each split. Additionally, trees are often pruned to increase

their predictive power. In bagging, each decision tree is grown from a corresponding bootstrap replicate $\mathbf{T}_{B,k}$ in stages (Breiman *et al.*, 1984). The results are then aggregated to give the bagging classifier (Section 2.3.5). The random forests method proceeds analogously to bagging except that only a random subset of the variables is considered at each (binary) split of each decision tree Breiman (2001). This strategic injection of randomness will lead to greater diversity amongst the trees, which in turn has been shown to decrease the misclassification rate of the random forest (Breiman, 2001).

2.3.5 Aggregation

After all decision trees are grown, the K learners must be combined to produce an aggregated classifier, $F_{\text{A}}(\mathbf{x})$, where the subscript "A" denotes aggregation. Random forests, as described by Breiman (1996a), use a majority vote when the target attribute is a class label. A majority vote for the classification of a new observation $\mathbf{x}$ is taken by calculating

$$N_g(\mathbf{x}) = \Big| \{F_k(\mathbf{x}) = g | k = 1, \ldots, K\} \Big|,$$
$$F_{\text{A}}(\mathbf{x}) = \underset{g}{\operatorname{argmax}}\, N_g(\mathbf{x}),$$

for $g = 1, \ldots, G$, where G is the number of class labels and $F_k(\mathbf{x})$ is the k^{th} tree classifier Breiman (1996a).

2.3.6 Out-of-Bag Estimates

After construction, there are two primary inferential objects that we concern ourselves with: (a) OOB data and (b) proximities. By using the OOB data we can estimate important quantities such as the generalization error (Breiman, 1996b). This generalization error estimate using the OOB data points is called the OOB error estimate. Breiman and Cutler (2004) describes that this estimate can be calculated while the ensemble is being trained, and requires only two steps:

1. For each observation $\mathbf{x}_i \in \mathbf{T} = \{(y_i, \mathbf{x}_i)|i = 1, \ldots, n\}$ that is OOB at least once, take a majority vote amongst the classifiers for which it is OOB.

2. Using the majority vote above and the true labels, calculate the misclassification rate of the OOB points, and this is the OOB error estimate.

This OOB error estimate has proven to be unbiased (Breiman, 2001) and nearly optimal (Breiman, 1996b) for the generalization error. Also, the use of the OOB error estimate is as accurate as using a test set equal in size to the training set (Breiman, 2001). Furthermore, the OOB data can be used to calculate other important quantities such as the number of random features sampled to create $\mathbf{T}_k$ and variable importance scores (Breiman, 2001).

2.3.7 Variable Importance

The concept of variable importance is described very well by Archer and Kimes (2008). Variable importance indicates the magnitude of the role of a particular variable in emulating the underlying mechanics behind the ensemble method. Variable

importance scores for a fixed number of predictors indicate, by relative size, which variables are important for classification. In other words, our goal in calculating variable importance scores is to gain insight into which covariates best contribute to the classification.

In random forests, Breiman (2001) outlines a procedure by which variable importance can be estimated using OOB data. This method has been slightly adapted herein to allow for the case when the predictors are FMMs. Consider the i^{th} feature vector $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,p})$. In order to calculate the raw importance score for the j^{th} variable $x_{i,j} = x_j$, perform the following steps:

1. Identify all predictors $F_k(\mathbf{x})$ which utilize x_j in their construction, along with the corresponding training data, $\mathbf{T}_k$.

2. Define $\mathbf{T}_{\text{oob},k} = \mathbf{T} \setminus \mathbf{T}_k$.

3. Predict the classes for $\mathbf{T}_{\text{oob},k}$ using F_k. Record the number of correct classifications.

4. Permute the values of x_j in each $\mathbf{T}_{\text{oob},k}$, to get $\mathbf{T}_{\text{perm},k}$. Repeat 3 using $\mathbf{T}_{\text{perm},k}$.

5. Subtract the number of correct classifications found using $\mathbf{T}_{\text{perm},k}$, from those using $\mathbf{T}_{\text{oob},k}$.

6. Average the differences from 5 over all predictors using x_j in their construction.

The average found in 6. is the estimated raw importance score for x_j. The intuition is that if the variable is important, then permuting the values will cause a large change in the predicted classifications. If the values of the raw importance scores

from predictor to predictor are independent, then we can treat the scores as being normally distributed according to the central limit theorem. The standard error can be calculated in the usual way, and each score can be divided by the standard error to give z-scores (Breiman and Cutler, 2004).

Variable importance can be used for tasks such as variable selection and this can help reduce the dimension of a high-dimensional data set. Dimensional reduction can be done to elucidate the underlying predictive structure or to decrease computation time without compromising the predictive power of the algorithm. Therefore, OOB data allow us to estimate generalization error and to estimate variable importance scores. However, this is just one of two important data objects generated by the random forests ensemble method. The other data object is proximities.

2.3.8 Proximities

From Breiman and Cutler (2004), for a dataset with N observations, the proximities of the observations are described by an $N \times N$ matrix $\mathbf{P}$. The entry in the ith row and the jth column, $P_{i,j}$ is the proximity between the ith and jth observation. After each model is trained, the entire dataset is classified and whenever cases $\mathbf{x}_i$ and $\mathbf{x}_j$ are placed in the same terminal node, the proximity measure between these two observations is increased by 1. At the end of this process, all proximities are normalized using the ensemble size. These proximity measures are useful for missing data and for outlier location.

Chapter 3

Mixture Discriminant Analysis Using Gaussian Mixture Models

3.1 Gaussian Mixture Models

MDA is a supervised classification technique where each class is represented by a finite mixture model. Recall that the finite mixture model density is

$$f(\mathbf{x}|\boldsymbol{\vartheta}) = \sum_{g=1}^{G} \pi_g f_g(\mathbf{x}|\boldsymbol{\theta}_g),$$

where $\pi_g > 0$ represents the gth mixing proportion, $\sum_{g=1}^{G} \pi_g = 1$, $\boldsymbol{\vartheta} = (\pi_1, \ldots, \pi_G, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_G)$ is the vector of parameters, $f_g(\mathbf{x}|\boldsymbol{\theta}_g)$ is the gth component density, and $f(\mathbf{x}|\boldsymbol{\vartheta})$ is then called the G component finite mixture density.

In traditional implementations of MDA (see, e.g., Fraley and Raftery, 2002), we

will constrain $f_g(\mathbf{x}|\boldsymbol{\theta}_g)$ to a multivariate Gaussian density:

$$f_g(\mathbf{x}|\boldsymbol{\theta}_g) = \frac{1}{\sqrt{(2\pi)^p\,|\boldsymbol{\Sigma}_g|}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_g)^\top \boldsymbol{\Sigma}_g^{-1}(\mathbf{x}-\boldsymbol{\mu}_g)\right\},$$

where $\boldsymbol{\theta}_g = (\boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g)$, $\boldsymbol{\mu}_g$ is the mean vector, and $\boldsymbol{\Sigma}_g$ is the covariance matrix. One problem with using the Gaussian mixture model in real applications is that there are $Gp(p+1)/2$ parameters in the component covariance matrices. Banfield and Raftery (1993) and Celeux and Govaert (1995) consider an eigen-decomposition of $\boldsymbol{\Sigma}_g$:

$$\boldsymbol{\Sigma}_g = \lambda_g \mathbf{D}_g \mathbf{A}_g \mathbf{D}_g^\top, \tag{3.1.1}$$

where $\lambda_g = |\boldsymbol{\Sigma}_g|^{\frac{1}{p}}$, $\mathbf{A}_g$ is a diagonal matrix of the normalized eigenvalues of $\boldsymbol{\Sigma}_g$ in decreasing order, and $\mathbf{D}_g$ is a matrix of associated eigenvectors matching the order in $\mathbf{A}_g$. The elements in the decomposition in (3.1.1) have a very useful interpretation. Specifically, the volume of the clusters is determined by λ_g, the orientation of the clusters is determined by $\mathbf{D}_g$, and the shape of the clusters is determined by $\mathbf{A}_g$. Further to this, as detailed in Celeux and Govaert (1995), by constraining λ_g, $\mathbf{D}_g$ and $\mathbf{A}_g$ the GPCM family is realized (Table 3.1). This family of models has advantages like tractability during parameter estimation, parsimony and interpretability.

As can be seen from the table, there are certain implicit assumptions when choosing the model (synonymous with "covariance structure"). McNicholas (2016) further describes the models in the family. The three types of models are spherical, diagonal and general. The general models in the GPCM family are flexible in their orientation, but the spherical and diagonal model types are not. Spherical and diagonal

Table 3.1: Characteristics of the members of the GPCM Family.

$\boldsymbol{\Sigma}_g$	Type	Model	Volume	Shape	Orientation
$\lambda\mathbf{I}$	Spherical	EII	Equal	Spherical	
$\lambda_g\mathbf{I}$	Spherical	VII	Variable	Spherical	
$\lambda\mathbf{A}$	Diagonal	EEI	Equal	Equal	Axis-Aligned
$\lambda_g\mathbf{A}$	Diagonal	VEI	Variable	Equal	Axis-Aligned
$\lambda\mathbf{A}_g$	Diagonal	EVI	Equal	Variable	Axis-aligned
$\lambda_g\mathbf{A}_g$	Diagonal	VVI	Variable	Variable	Axis-aligned
$\lambda\mathbf{DAD}^\top$	General	EEE	Equal	Equal	Equal
$\lambda_g\mathbf{DAD}^\top$	General	VEE	Variable	Equal	Equal
$\lambda\mathbf{DA}_g\mathbf{D}^\top$	General	EVE	Equal	Variable	Equal
$\lambda\mathbf{D}_g\mathbf{AD}_g^\top$	General	EEV	Equal	Equal	Variable
$\lambda_g\mathbf{DA}_g\mathbf{D}^\top$	General	VVE	Variable	Variable	Equal
$\lambda_g\mathbf{D}_g\mathbf{AD}_g^\top$	General	VEV	Variable	Equal	Variable
$\lambda\mathbf{D}_g\mathbf{A}_g\mathbf{D}_g^\top$	General	EVV	Equal	Variable	Variable
$\lambda_g\mathbf{D}_g\mathbf{A}_g\mathbf{D}_g^\top$	General	VVV	Variable	Variable	Variable

models implicitly assume that the variables are independent, resulting in their lack of flexibility in their orientation. On the other hand, the advantage of spherical and diagonal models is their parsimony.

3.2 MDA Framework

In MDA, a Gaussian mixture model is fit for each known class. Therefore, in effect, a model-based clustering analysis is carried out for each known class. Each classifier, $F_k(\mathbf{x})$, is fitted/trained using maximum likelihood estimation via an expectation-maximization (EM) algorithm (Dempster *et al.*, 1977). Though our primary interest is in MDA, we are effectively employing model-based clustering and so parameter estimation for both is discussed.

A prerequisite for understanding MDA is the definition of the component memberships. While it is our goal to use the mixture model to estimate the component memberships, it factors into parameter estimation — this is well covered in McNicholas (2016), which is the basis for the remainder of this section. In a G component mixture model, we define the component membership vector of the ith observation to be $\mathbf{z}_i = (z_{i,1}, \ldots, z_{i,G})$, where $z_{i,g} = 1$ if $\mathbf{x}_i$ is in the gth component, and $z_{i,g} = 0$ otherwise. According to McLachlan and Peel (2000a), for independent feature data, $\mathbf{Z}_1, \ldots, \mathbf{Z}_N$ are distributed as i.i.d. multinomial random variables with parameters 1 and $\boldsymbol{\pi}$. We have the following support: $z_{i1}, \ldots, z_{iG} \in \{0, 1\}$ and $\sum_{i=1}^{G} z_{ig} = 1$. The probability function is

$$P(\mathbf{Z}_i = \mathbf{z}_i) = \prod_{j=1}^{G} \frac{\pi_j^{z_{ij}}}{z_{ij}!}$$

and the *a posteriori* probability given $\mathbf{x}_i$ is

$$P(Z_{ig} = 1 \mid \mathbf{x}_i) = \frac{\pi_g f_g(\mathbf{x}_i|\boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g)}{\sum_{j=1}^{G} \pi_j f_j(\mathbf{x}_i|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)},$$

which is also equal to the posterior expected value, i.e., $E(Z_{ig} \mid \mathbf{x}_i) = P(Z_{ig} = 1 \mid \mathbf{x}_i)$. As such, a natural estimator for the z_{ig}, once the parameters have been estimated, is

$$\hat{z}_{ig} = \frac{\hat{\pi}_g f_g(\mathbf{x}_i|\hat{\boldsymbol{\mu}}_g, \hat{\boldsymbol{\Sigma}}_g)}{\sum_{j=1}^{G} \hat{\pi}_j f_j(\mathbf{x}_i|\hat{\boldsymbol{\mu}}_j, \hat{\boldsymbol{\Sigma}}_j)}. \tag{3.2.1}$$

After the z_{ig} are estimated, it can be seen that $\hat{z}_{ig} \in [0, 1]$ since they are probabilities of belonging to a component. These are known as soft classifications, and can be

hardened into maximum a posteriori classifications (i.e., MAP$\{\hat{z}_{ig}\}$), such that

$$\text{MAP}\{\hat{z}_{ig}\} = \begin{cases} 1, & \text{if } g = \text{argmax}_h\{\hat{z}_{ih}\}, \\ 0, & \text{otherwise.} \end{cases}$$

For the model-based clustering case, where we have not observed any class labels, the observed data is comprised of the following set of feature vectors: $\mathbf{T} = \{\mathbf{x}_i | i = 1, \ldots, N\}$. The likelihood equation is

$$\mathscr{L}(\boldsymbol{\vartheta}) = \prod_{i=1}^{N} \sum_{g=1}^{G} \pi_g f_g(\mathbf{x}_i | \boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g),$$

where $\boldsymbol{\vartheta} = (\boldsymbol{\pi}, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_G, \boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_G)$ is the vector of parameters. Because we do not know the component memberships of the observations, we are dealing with an incomplete data problem. If the component memberships are known, then the complete-data can be written as $\mathbf{D} = \{(\mathbf{z}_i, \mathbf{x}_i) | i = 1, \ldots, N\} = (\mathbf{Z}, \mathbf{X})$, where $\mathbf{Z} = (\mathbf{Z}_1, \ldots, \mathbf{Z}_G)$. This allows us to write the complete-data likelihood equation for model-based clustering, as follows:

$$\mathscr{L}_c(\boldsymbol{\vartheta}) = \prod_{i=1}^{N} \prod_{g=1}^{G} \left[\pi_g f_g(\mathbf{x}_i | \boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g)\right]^{z_{ig}}.$$

Now, for MDA, we account for the fact that within each observed class there may be sub-classes by performing model-based clustering within each class g. There will be $G_g \geq 1$ components in each class, making the number of model components $H = G_1 + \cdots + G_G$. Then the likelihood equation for MDA, using only the n

observations for which we have an observed label, is

$$\mathscr{L}(\boldsymbol{\vartheta}) = \prod_{i=1}^{n} \prod_{h=1}^{H} \left[\pi_h f_h(\mathbf{x}_i|\boldsymbol{\mu}_h, \boldsymbol{\Sigma}_h)\right]^{z_{ih}} . \tag{3.2.2}$$

There are potential difficulties in maximizing these equations with respect to $\boldsymbol{\vartheta}$. As described by Titterington *et al.* (1985), for simple parametric models the estimates are often easy to compute, but for mixture models the asymptotic theory and computational issues are more difficult. Titterington further describes that even in the most commonly applied mixture of two univariate Gaussian densities the resulting likelihood surface is pitted with singularities. Thus, in Section 3.3, we turn to a popular iterative algorithm in order to compute the maximum likelihood estimates.

3.3 EM Algorithm

The EM algorithm is a general iterative method of finding the maximum likelihood estimates (MLEs) of the parameters in the underlying distribution of observed random variables. Specifically, it is used to handle cases when there is unobserved or missing data, making it useful in handling the z_{ig}. For the purposes of this paper, the model fitting was performed using the `mclust` package in R (Scrucca *et al.*, 2016). The Initialization step of the EM is a critical step. While the EM algorithm is easy to implement, numerically stable and has desirable convergence properties, there are some issues. The likelihood surface from the mixture model can have singularities and many local maxima. This means the parameter estimation depends heavily on the starting point of the algorithm.

In the `mclust` package, the EM algorithm is initialized using the partitions resulting from model-based hierarchical agglomerative clustering. In this clustering method, no a priori information is assumed, and clusters are chosen by merging observations together based on the minimum decrease in the classification likelihood function. A detailed explanation is given in Fraley and Raftery (2002). The nice part of this initialization technique is that the same underlying model is used in both this step and the model fitting step. While this initialization technique has some good properties, it doesn't always result in the global maximum being chosen. There are a number of ways to initialize the EM Algorithm, similar to deterministic annealing which is discussed in McNicholas (2016) and Zhou and Lange (2010).

As discussed by McLachlan and Peel (2000a) and McNicholas (2016), the EM algorithm is usually stopped when from one iteration to the next, the parameter estimates or the log-likelihood exhibit very little change. When the log-likelihood is considered, the EM algorithm is stopped when the log-likelihood at the kth step $l^{(k)} = \log L(\mathbf{v}^{(k)})$ is only a little smaller than the log-likelihood at the $k+1$th step $l^{(k+1)}$ by a predetermined threshold, i.e.,

$$l^{(k+1)} - l^{(k)} < \epsilon$$

for ϵ small. But, this only captures the lack of progress made by the algorithm. When convergence is not smooth this might lead to prematurely stopping the EM algorithm, so we can also consider criteria based on Aitken's acceleration:

$$a^{(k)} = \frac{l^{(k+1)} - l^{(k)}}{l^{(k)} - l^{(k-1)}},$$

which is the acceleration at the kth step. If it is assumed that the log-likelihood will eventually converge (which is reasonable), then the asymptotic log-likelihood at the $k+1$th step is

$$l_\infty^{(k+1)} = l^{(k)} + \frac{l^{(k+1)} - l^{(k)}}{1 - a^{(k)}}$$

and one version of a stopping criterion is

$$l_\infty^{(k+1)} - l^{(k)} < \epsilon$$

for ϵ small, if this difference is positive (McNicholas *et al.*, 2010). For alternative stopping rules see McNicholas (2016).

3.3.1 EM Algorithm for MDA

Beginning with initial parameter estimates, $\boldsymbol{\vartheta}^{(0)}$, we iterate between two steps:

1. Expectation (E-) step: Compute

$$Q(\boldsymbol{\vartheta}|\hat{\boldsymbol{\vartheta}}^{(i-1)}) = E\left[\log \mathscr{L}_c(\boldsymbol{\vartheta}|\mathbf{x},\mathbf{z})\middle|\mathbf{x},\hat{\boldsymbol{\vartheta}}^{(i-1)}\right], \tag{3.3.1}$$

   where $\mathbf{x}$ is the matrix of observed feature vectors and $\mathbf{z}$ is the matrix of associated component memberships.

2. Maximization (M-) step: Choose $\hat{\boldsymbol{\vartheta}}^{(i)}$ to be a value of $\boldsymbol{\vartheta}$ belonging to the parameter space and maximizing $Q(\boldsymbol{\vartheta}|\hat{\boldsymbol{\vartheta}}^{(i-1)})$.

This assumes that we begin the EM algorithm with an initial estimate of the parameter values, denoted $\boldsymbol{\vartheta}^{(0)}$. Thus, we need to consider where to start the algorithm and when to stop it.

The specific EM algorithm defined for both model-based clustering case and MDA are clearly detailed in McNicholas (2016). For the model-based clustering case, the E-step is defined by calculating $Q(\boldsymbol{\vartheta}|\hat{\boldsymbol{\vartheta}}^{(i-1)})$:

$$\begin{aligned} Q(\boldsymbol{\vartheta}|\hat{\boldsymbol{\vartheta}}^{(i-1)}) &= E\left[\log \mathscr{L}_c(\boldsymbol{\vartheta}|\mathbf{X},\mathbf{Z})\Big|\mathbf{X},\hat{\boldsymbol{\vartheta}}^{(i-1)}\right], \\ &= \sum_{i=1}^{N}\sum_{g=1}^{G} E(z_{ig}|\mathbf{X},\hat{\boldsymbol{\vartheta}}^{(i-1)}) \log \pi_g \\ &\qquad + \sum_{i=1}^{N}\sum_{g=1}^{G} E(z_{ig}|\mathbf{X},\hat{\boldsymbol{\vartheta}}^{(i-1)}) \log f_g(\mathbf{x}_i|\boldsymbol{\mu}_g,\boldsymbol{\Sigma}_g), \\ &= \sum_{i=1}^{N}\sum_{g=1}^{G} \hat{z}_{ig} \log \pi_g + \sum_{i=1}^{N}\sum_{g=1}^{G} \hat{z}_{ig} \log f_g(\mathbf{x}_i|\boldsymbol{\mu}_g,\boldsymbol{\Sigma}_g), \\ &= \sum_{g=1}^{G} n_g \log \pi_g - \frac{Np}{2}\log(2\pi) - \sum_{g=1}^{G}\frac{n_g}{2}\log|\boldsymbol{\Sigma}_g| - \sum_{g=1}^{G}\frac{n_g}{2}\operatorname{tr}\left\{\mathbf{S}_g\boldsymbol{\Sigma}_g^{-1}\right\}, \end{aligned} \tag{3.3.2}$$

where $n_g = \sum_{i=1}^{N} \hat{z}_{ig}$ and

$$\mathbf{S}_g = \frac{1}{n_g}\sum_{i=1}^{N} \hat{z}_{ig}\left(\mathbf{x}_i - \boldsymbol{\mu}_g\right)\left(\mathbf{x}_i - \boldsymbol{\mu}_g\right)^{\top}.$$

In (3.3.2), the complete-data log-likelihood is linear in the z_{ig}, so computing $Q(\boldsymbol{\vartheta}|\hat{\boldsymbol{\vartheta}}^{(i-1)})$ is the same as replacing the z_{ig}'s with their expected values (McLachlan and Peel, 2000a). As discussed above, given the parameter estimates from the previous EM iteration, we can use (3.2.1) to do this.

Then, in the M-step, we maximize $Q(\boldsymbol{\vartheta}|\hat{\boldsymbol{\vartheta}}^{(i-1)})$, i.e., (3.3.1), with respect to the parameters π_g, $\boldsymbol{\mu}_g$, and $\boldsymbol{\Sigma}_g$. This yields the following parameter estimates:

$$\begin{aligned}
\hat{\pi}_g &= \frac{n_g}{N}, \\
\hat{\boldsymbol{\mu}}_g &= \frac{1}{n_g}\sum_{i=1}^{N}\hat{z}_{ig}\mathbf{x}_i, \\
\boldsymbol{\Sigma}_g &= \frac{1}{n_g}\sum_{i=1}^{N}\hat{z}_{ig}(\mathbf{x}_i-\hat{\boldsymbol{\mu}}_g)(\mathbf{x}_i-\hat{\boldsymbol{\mu}}_g)^\top.
\end{aligned}$$

The EM algorithm for Gaussian mixture models consists of a set of initial estimates $\left\{\hat{\pi}_g^{(0)}, \hat{\boldsymbol{\mu}}_g^{(0)}, \hat{\boldsymbol{\Sigma}}_g^{(0)} \middle| g = 1, \ldots, G\right\}$ and then performing the E-step and M-step, updating the parameter estimates until the convergence criterion is met.

In MDA, we use only the labelled observations to perform our inference. There are N total observations: n labelled, and $N-n$ unlabelled. Without loss of generality we can write our observations as follows: $\mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{x}_{n+1}, \ldots, \mathbf{x}_N$, placing the labelled observations first.

The one-to-one correspondence between model components and observed classes can be relaxed, and likelihood (3.2.2) can be used. When deciding the number of components, it is necessary to use model selection criteria (more on this later). Otherwise, the E-step and the M-step are very similar to those used for MBC.

The E-step requires the calculation of $Q(\boldsymbol{\vartheta}|\hat{\boldsymbol{\vartheta}}^{(i-1)})$:

$$Q(\boldsymbol{\vartheta}|\hat{\boldsymbol{\vartheta}}^{(i-1)}) = \sum_{h=1}^{H} n_h \log \pi_h - \frac{np}{2}\log(2\pi) - \sum_{h=1}^{H} \frac{n_h}{2}\log|\boldsymbol{\Sigma}_h| - \sum_{h=1}^{H} \frac{n_h}{2}\operatorname{tr}\left\{\mathbf{S}_h\boldsymbol{\Sigma}_h^{-1}\right\}.$$

For the M-step we maximize with respect to the parameters, yielding:

$$\begin{aligned}
\hat{\pi}_h &= \frac{n_h}{N}, \\
\hat{\boldsymbol{\mu}}_h &= \frac{1}{n_h}\sum_{i=1}^{n} z_{ih}\mathbf{x}_i, \\
\hat{\boldsymbol{\Sigma}}_h &= \frac{1}{n_h}\sum_{i=1}^{n} z_{ih}\left(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_h\right)\left(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_h\right)^{\top}.
\end{aligned}$$

Component memberships for the remaining $N - n$ unlabelled observations are estimated as follows:

$$\hat{z}_{ig} = \frac{\hat{\pi}_g f_g(\mathbf{x}_i|\hat{\boldsymbol{\mu}}_g, \hat{\boldsymbol{\Sigma}}_g)}{\sum_{h=1}^{H}\hat{\pi}_h f_h(\mathbf{x}_i|\hat{\pi}_h, \hat{\boldsymbol{\Sigma}}_h)},$$

for $i = n+1, \ldots, N$ and $g = 1, \ldots, H$.

Therefore, in order to perform the MDA the following steps/tasks are performed:

1. Cluster the Observations in Each Observed Class

   For each $g \in \{1, \ldots, G\}$, a model-based clustering is done whereby a G_g component mixture is selected for each class according to the BIC, and the associated labels are observed.

2. Fit the Mixture Model

Fit a model with $H = \sum_{g=1}^{G} G_i$ components to the observations $\mathbf{x}_1, \ldots, \mathbf{x}_n$ using the group labels recorded in the previous step, and calculate the parameter estimates: $\hat{\boldsymbol{\pi}}$, $\hat{\mu}_1, \ldots, \hat{\mu}_H$, and $\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_H$.

3. Estimate Component Memberships of Unlabelled Observations

   For $j = k+1, \ldots, N$, calculate $\hat{z}_{jg}$ for each component in the mixture model from the previous step, and assign $\mathbf{x}_j$ to the class for which it's $\hat{z}_{jg}$ is maximum.

3.4 Bayesian Information Criterion (BIC)

A central question in the FMM context, is the selection of the number of components. There are a number of formal hypothesis testing approaches, but information criteria are in widespread use, especially the Bayesian information criterion (BIC; Schwarz, 1978). The BIC is given by

$$\text{BIC}_{\mathcal{M},G} = 2\ell_{\mathcal{M},G}(\hat{\boldsymbol{\vartheta}}|\mathbf{x}) - \rho \log n, \tag{3.4.1}$$

where $\mathcal{M}$ is the model, G is the number of components, ρ is the number of free parameters, and N is the number of observations. The pair $\{\mathcal{M}, G\}$ that maximizes (3.4.1) is chosen.

Essentially, information criteria are penalized forms of the log-likelihood. The log-likelihood increases as G (the number of components) increases, so the BIC is penalized for the number of parameters needed for the particular FMM. Otherwise, the most complex model would always be selected. The relationship of the BIC with

Bayes factors is particularly important in mixture model selection (see McNicholas, 2016).

Justification for the BIC comes from the framework of Bayesian model selection, specifically the use of the quantity known as the Bayes factor. Details for the use of the Bayes factor can be found in Raftery (1995). Supposing that the observed data ($\mathbf{X}$) is likely, two competing models $\mathcal{M}_1$ and $\mathcal{M}_2$, with parameters $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$, can be compared using Bayes' theorem to calculate the posterior probability that $\mathcal{M}_1$ is the correct model:

$$\Pr(\mathcal{M}_1 \mid \mathbf{X}) = \frac{\Pr(\mathbf{X} \mid \mathcal{M}_1)\Pr(\mathcal{M}_1)}{\Pr(\mathbf{X} \mid \mathcal{M}_1)\Pr(\mathcal{M}_1) + \Pr(\mathbf{X} \mid \mathcal{M}_2)\Pr(\mathcal{M}_2)},$$

provided that one of them is indeed the correct model. The marginal probability of $\mathbf{X}$ given model k ($\mathcal{M}_k$), for $k = 1, 2$ is given by

$$\Pr(\mathbf{X} \mid \mathcal{M}_k) = \int_{\boldsymbol{\theta} \in \boldsymbol{\Omega}} \Pr(\mathbf{X} \mid \boldsymbol{\theta}_k, \mathcal{M}_k)\Pr(\boldsymbol{\theta}_k \mid \mathcal{M}_k)\, d\boldsymbol{\theta}_k. \tag{3.4.2}$$

The posterior odds can also be used to determine which of model $\mathcal{M}_1$ or $\mathcal{M}_2$ is more suitable according to the observed data $\mathbf{X}$. The posterior odds is given by

$$\frac{\Pr(\mathcal{M}_2 \mid \mathbf{X})}{\Pr(\mathcal{M}_1 \mid \mathbf{X})} = \left[\frac{\Pr(\mathbf{X} \mid \mathcal{M}_2)}{\Pr(\mathbf{X} \mid \mathcal{M}_1)}\right] \times \left[\frac{\Pr(\mathcal{M}_2)}{\Pr(\mathcal{M}_1)}\right].$$

That is,

$$\text{posterior odds} = \text{Bayes factor} \times \text{prior odds}.$$

In the case where $\mathcal{M}_1$ and $\mathcal{M}_2$ are equally likely, the prior odds are equal to 1.

This means that the Bayes factor, denoted B_{21}, is the main inferential quantity in determining whether or not the data supports $\mathcal{M}_2$ over $\mathcal{M}_1$ or *vice versa*. In Appendix B of Jeffreys (1961), there are guidelines which outline how much evidence is constituted by B_{21} based on its value.

The main issue with using Bayes factors as an inferential quantity is the calculation of the marginal likelihood given by (3.4.2). In high dimensions the marginal likelihood may not be easily solved, having no closed form solution. Thus, an approximation to the Bayes factor is required.

Raftery (1995) shows that under certain conditions the BIC is the approximation to $2 \log B_{21}$, where B_{21} is the Bayes factor. Meaning, the BIC gives evidence to support the use of one model over another according to the Bayesian model selection framework.

3.4.1 Performance Assessment

There is sometimes a need to compare two partitions of the same data points. This can be a way of comparing the clustering results from one model with those of another model. It can also be a way of comparing the clustering that results from fitting a model with a priori class labels, i.e., comparing clustering labels with class labels. The Rand index (RI; Rand, 1971) and the adjusted Rand index (ARI; Hubert and Arabie, 1985) are measures of how well two partitions agree with one another. According to Fern (2012), we define two partitions P_1 and P_2, and the following variables:

- Let A represent the number of vector pairs that are in the same class in both

P_1 and P_2.

- Let B represent the number of vector pairs that are in the same class in P_1, but in different classes in P_2.
- Let C represent the number of vector pairs that are in different classes in P_1, but are in the same class in P_2.
- Let D represent the number of vector pairs that are in different classes in P_1 and P_2.

McNicholas (2016) gives formulae, using the notation in Steinley (2004), as follows:

$$\text{RI} = \frac{A+D}{N},$$

$$\text{ARI} = \frac{N(A+D) - [(A+B)(A+C)+(C+D)(B+D)]}{N^2 - [(A+B)(A+C)+(C+D)(B+D)]},$$

where $N = A+B+C+D$. The adjustment in the ARI takes into account any chance agreements. McNicholas (2016) further explains that under random class assignment the ARI has an expected value of 0, a maximum value of 1 indicating perfect class agreement, and can be negative indicating worse than random classification.

Another way to assess model performance, as in to test classification accuracy, is to perform K-fold cross validation. This involves randomly partitioning the data into K equally sized subsets $\{D_1, \ldots, D_K\}$. The model is fit K times, leaving out one D_k, for $k = 1, \ldots, K$, each time. This left out set is called the validation set and the remaining sets used to fit the model are considered together as the training set.

As the name suggests, the models are validated using their D_k and the classification accuracy is gauged.

3.5 MixForests

3.5.1 Random Feature Selection

There is no mixture modelling analogue of a "split" in a decision tree; accordingly, our random forests analogue (MixForests) employs random feature selection exactly once in training each learner. In this scheme, each model in the analogue ensemble will be fitted to a random subset of the data in its corresponding bootstrap replicate $\mathbf{T}_{B,k}$. Before each model is trained, r features are randomly chosen, and $\mathbf{T}_k$ is created, i.e., $\mathbf{T}_{B,k} \mapsto \mathbf{T}_k$.

As an example, take a feature vector $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,6})'$ consisting of six features and $r = 3$. One possible outcome of the random feature selection might be $\mathbf{x}_i^{\text{new}} = (x_{i3}, x_{i1}, x_{i5})$. So, only the first, third and fifth features would be used to train $F_k(\cdot)$. Every time a new learner is trained, a random selection of features is taken from its corresponding $\mathbf{T}_{B,k}$. Then, we understand $F_k(\mathbf{x}, \mathbf{T}_k)$ to mean $F_k(\mathbf{x})$ and our ensemble is $\{F_k(\mathbf{x}, \mathbf{T}_k)|k = 1, \ldots, K\} = \{F_k(\mathbf{x})|k = 1, \ldots, K\}$.

3.5.2 Aggregation: 2 Types

For MixForests, there are two options for aggregation. The first option is to do a majority vote as described in Section 2.3.5. The second option is to aggregate our analogue ensemble using a direct averaging of the z_{ig} that are estimated during

model fitting. However, there is a small complication with the second approach. As described in Section 3.2, when fitting a Gaussian mixture model many components can be fitted to one class. This produces an ensemble where the learners have a varying number of components. So, the number of components must be decided and the models in the ensemble must either be discarded or undergo *a posteriori* merging.

To select the number of components, we first categorize the models according to the number of components. We can either use the smallest number of components, or we can use the number of components in the largest subset of models. Either way, the reference model will be taken as the model with the largest BIC in the category of interest. When the number of components is decided, call it g_A, we proceed as follows:

(a) discard the models with fewer than g_A components;

(b) keep the models with exactly g_A components; and

(c) perform a posteriori merging of components for models which have more than g_A components.

In the last case, *a posteriori* merging of components is given by an algorithm laid out in Wei and McNicholas (2015). As in their description, the process is best laid out as an example.

Example

Suppose it is our goal to perform a posteriori merging on a model with 5 components, so that it will have only 3 components. Set $g_{\text{A}} = 3$ and $g_{\text{new}} = 5$. Then we perform

the following steps.

1. Create a Combination Matrix $\mathbf{A}$

   The matrix $\mathbf{A}$ will have dimensions $\binom{g_{\text{new}}}{g_{\text{A}}} \times g_{\text{A}}$. Meaning, $\mathbf{A}$ will have dimensions 10×3. The rows will represent all possible subsets of size 3, of the components in the model, and will give an indication of a mapping to a 3 component model. The matrix $\mathbf{A}$ will be as follows.

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 4 \\ 1 & 2 & 5 \\ 1 & 3 & 4 \\ 1 & 3 & 5 \\ 1 & 4 & 5 \\ 2 & 3 & 4 \\ 2 & 3 & 5 \\ 2 & 4 & 5 \\ 3 & 4 & 5 \end{bmatrix}.$$

   The seventh row, $a_7 = (2, 3, 4)$, indicates the mappings: $2 \mapsto 1$, $3 \mapsto 2$, and $4 \mapsto 3$.

2. Create Matrix $\mathbf{A}^c$ and Permutation Matrix $\mathbf{B}$

   The matrix $\mathbf{A}^c$ will represent the components that were not selected in $\mathbf{A}$ and will have dimensions $\binom{g_{\text{new}}}{g_A} \times (g_{\text{new}} - g_A) = 10 \times 2$. The permutation matrix $\mathbf{B}$

will have dimensions $g_A^{g_{\text{new}}-g_A} \times (g_{\text{new}} - g_A) = 9 \times 2$.

$$\mathbf{A}^c = \begin{bmatrix} 4 & 5 \\ 3 & 5 \\ 3 & 4 \\ 2 & 5 \\ 2 & 4 \\ 2 & 3 \\ 1 & 5 \\ 1 & 4 \\ 1 & 3 \\ 1 & 2 \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 1 & 2 \\ 2 & 2 \\ 3 & 2 \\ 1 & 3 \\ 2 & 3 \\ 3 & 3 \end{bmatrix}.$$

The elements in $\mathbf{A}^c$ are mapped according to each row of $\mathbf{B}$. For example, take the first row of $\mathbf{A}^c$. According to row 1 of $\mathbf{B}$, the mapping is $4 \mapsto 1$ and $5 \mapsto 1$. According to row 2 of $\mathbf{B}$, the mapping is $4 \mapsto 2$ and $5 \mapsto 1$. Etc.

3. Create matrix $\mathbf{C}$

   A new partition will result from a combination of rows from $\mathbf{A}$ and $\mathbf{B}$. So, this method generates $\binom{g_{\text{new}}}{g_A} \times g_A^{g_{\text{new}}-g_A} = 90$ new partitions.

   The ARIs between the reference partition (coming from the reference model) and these newly calculated partitions will be recorded in a matrix $\mathbf{C}$, with dimensions $\binom{g_{\text{new}}}{g_A} \times g_A^{g_{\text{new}}-g_A}$. The merging of components will take place according to the entry with the largest ARI.

After all the models in the analogue ensemble have been discarded or have undergone a posteriori merging, the z_{ig} can be easily averaged. This should produce results similar to those of a majority vote, but this will have to be tested empirically.

3.5.3 Proximities

From Breiman and Cutler (2004), for a dataset with N observations, the proximities of the observations are described by an $N \times N$ matrix P. The entry in the ith row and the jth column, $P_{i,j}$, is the proximity between the ith and jth observation. After each model is trained, the entire dataset is classified and whenever cases $\mathbf{x}_i$ and $\mathbf{x}_j$ are classified/placed in the same component in any given model, the proximity measure between these two observations is increased by 1. At the end of this process, all proximities are normalized using the ensemble size. These proximity measures are useful, *inter alia*, for missing data and for outlier location.

3.6 Data Analyses and Illustrations

3.6.1 Training and Test Sets

For each analysis herein, the data sets were partitioned into a training set and test set. Each test set was constructed using stratified sampling, i.e., by randomly sampling 20% of the data points belonging to each class, rather than randomly sampling 20% of the entire data set.

3.6.2 Pima Indians Diabetes Data

The original data came from the National Institute of Diabetes and Digestive and Kidney Diseases. For more information see Smith *et al.* (1988). The version of data used is the corrected version found at the UCI Machine Learning Repository (Lichman, 2013). The data consists of 768 observations made on Pima Indian women, with 9 variables. The training set consists of 614 observations and the test set consists of 154 observations. The variables are as follows.

1. *diabetes*: class variable, test for diabetes.
2. *pregnant*: number of times pregnant.
3. *glucose*: plasma glucose concentration.
4. *pressure*: diastolic blood pressure in $mmHg$.
5. *triceps*: triceps skin fold thickness in mm.
6. *insulin*: 2-Hour serum insulin in $\mu U/ml$.
7. *mass*: body mass index.
8. *pedigree*: diabetes pedigree function.
9. *age*: in years.

From the pairs plot (Figure 3.1) it is not evident that there are any clear decision boundaries, so we cannot move forward with any particular intuition regarding relationships between the variables.

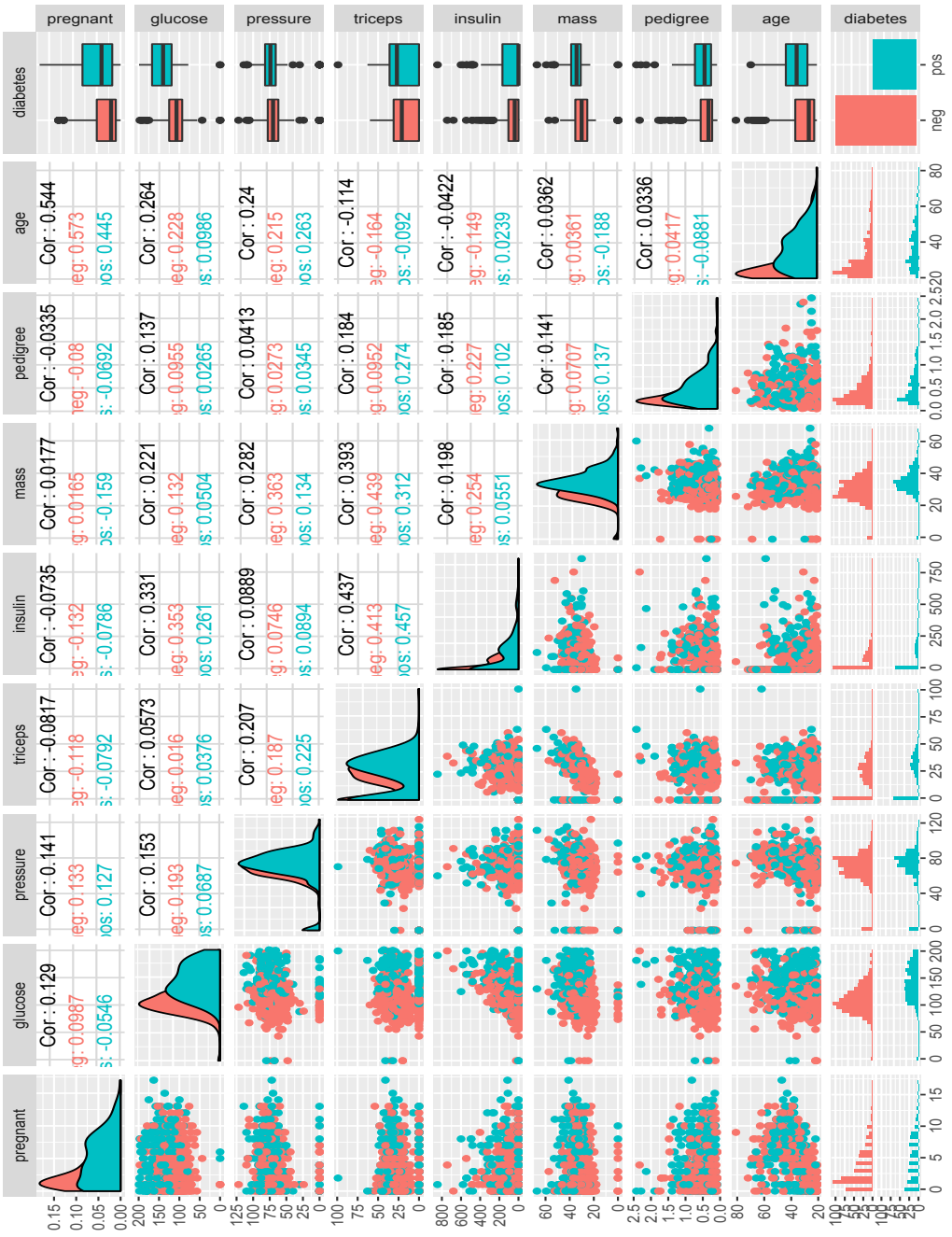

Figure 3.1: A pairs plot of the Pima Indians Diabetes data.

Now we consider the construction of the analogue ensemble, specifically which value of r should be used to make inferences. We fitted up to $K = 5000$ models for each choice of r and graphed the results (Figure 3.2). It is important to note that convergence in the OOB error is achieved for a much smaller choice of ensemble size K.

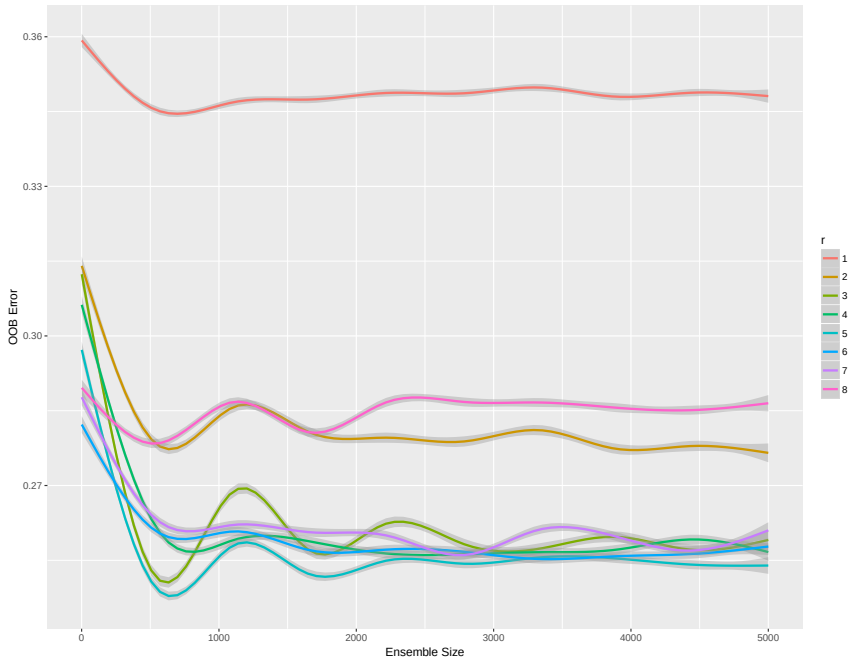

Figure 3.2: OOB error convergence for Pima Indians diabetes data for different values of ensemble size and r.

Taking a look at the final values, we can examine the best choice of r more closely (Table 3.2). There appears to be a sweet spot for the choice of r, though we have a wide range of reasonable choices. For the time being, we base our inference on a choice of $r = 5$ because the final error and minimum error are the lowest.

Table 3.2: Final and minimum OOB errors for different values of r.

r	Final OOB error	Minimum OOB error
1	0.349	0.340
2	0.275	0.269
3	0.259	0.249
4	0.257	0.249
5	0.254	0.244
6	0.256	0.252
7	0.261	0.254
8	0.287	0.264

Now, we take a look at the variable importance scores using $r = 5$ (Table 3.3). Unsurprisingly, the variable *glucose* is the most relevant, followed by *age* and *mass* (to a lesser degree). It is possible to fit a model with a reduced data set for greater parsimony using these results.

Table 3.3: Variable importance for the MixForests analysis of the diabetes data.

Variable	Raw score	z-score	p-value
pregnant	6.009	0.890	0.373
glucose	21.319	2.691	0.007
pressure	0.635	0.114	0.909
triceps	2.594	0.344	0.731
insulin	2.278	0.325	0.745
mass	6.367	0.984	0.325
pedigree	3.620	0.600	0.549
age	10.841	1.374	0.169

Applying MixForests to the test set, we have error rates between 0.227 and 0.344 (Table 3.4). We now compare the performance of mixForest and the mixture discriminant analysis, in terms of classification accuracy, using the test set. The test error for mixture discriminant analysis is 0.234. If we compare this to our choice of $r = 5$, we see that mixture discriminant analysis performs slightly better. However,

for a choice of $r = 4$ we have the same performance and for $r = 6$ we have slightly better performance. For completeness, the confusion matrix for mixForest for $r = 6$ is given in Table 3.5 .

Table 3.4: Test errors for the mixForest method.

r	Test error
1	0.344
2	0.299
3	0.253
4	0.234
5	0.240
6	0.227
7	0.247
8	0.247

Table 3.5: Cross-tabulation of true versus predicted classifications for $r = 6$, where overlines denote predictions.

	$\overline{\text{Counterfeit}}$	$\overline{\text{Genuine}}$
Counterfeit	88	12
Genuine	23	31

3.6.3 Wireless Indoor Localization Data

The original data came from a study (Rohra *et al.*, 2017) that attempted user localization in indoor environments using cellular phone signals. The data consists of seven wireless signals taken from handsets, labelled: *WS1*, *WS2*, *WS3*, *WS4*, *WS5*, *WS6*, *WS7*. The corresponding responses/labels indicate the location of the handset in one of four zones. There are a total of 1999 observations split into a training set of size 1599 and a test set of size 400. Our first look at the data is a simple pairs plot,

to see if any correlation or patterning is visually identifiable (Figure 3.3). There is clear patterning, especially when examining the box plots.

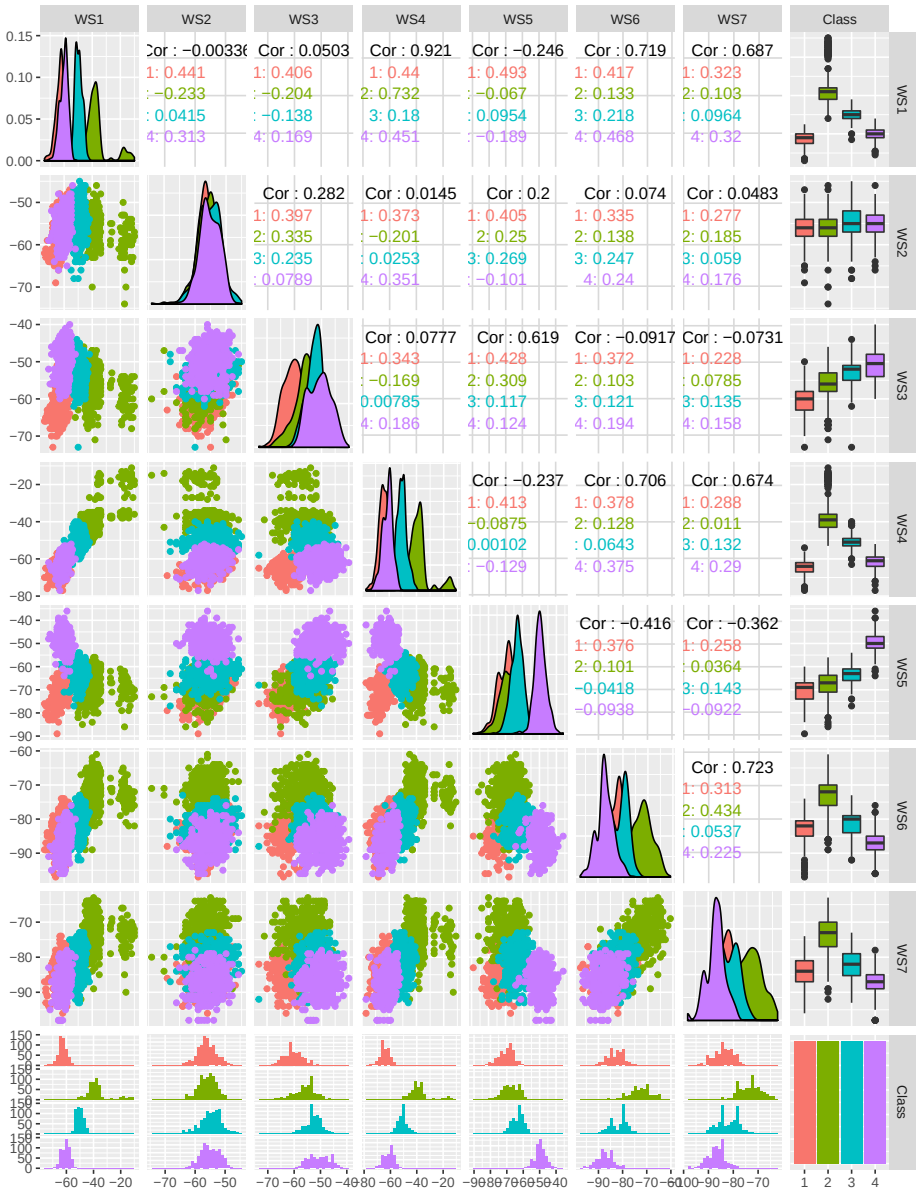

Figure 3.3: A pairs plot of the wireless indoor localization data.

Next, we consider the number of randomly selected features r. There should be

a range of values of r which will lead to superior inferences about the data. To do this, ensembles of up to $K = 5000$ were fitted for $r = 1, \ldots, 7$. We can see that for higher values of r we achieve better results (Figure 3.4).

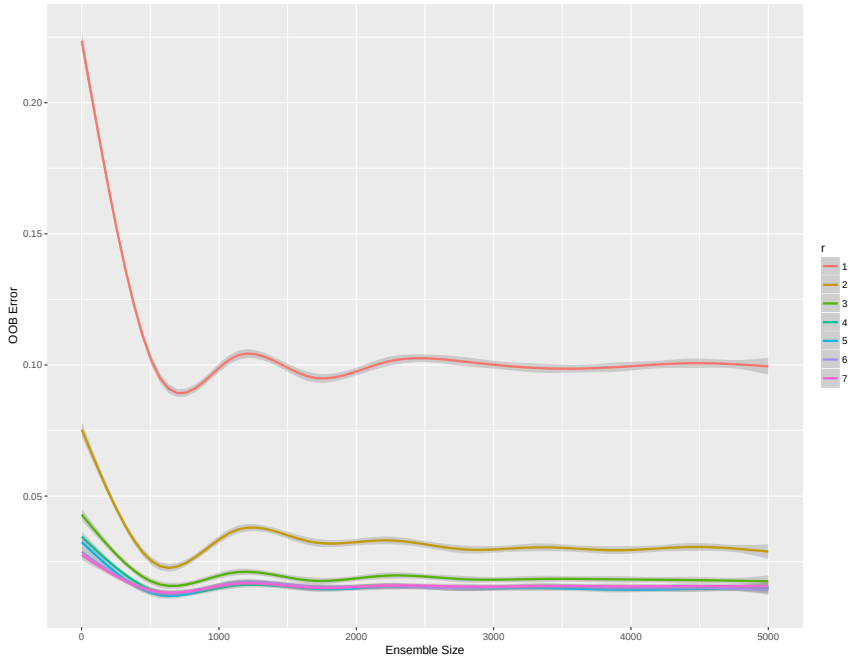

Figure 3.4: OOB error plot for the wireless indoor localization data.

Taking a look more closely at the exact values in the numerical chart of the OOB error series produced by each ensemble (Table 3.6), we see that we get similar performance for $r > 4$. However, the optimal choice appears to be $r = 5$, which yields the lowest minimum value for OOB error as well as the lowest final value. As far as convergence, an ensemble size of $K = 5000$ seems excessive, as convergence appears to happen rapidly. An ensemble of $K = 1000$ appears to be more than sufficient. From here on, we will use $r = 5$ to perform any further inferences.

From here, we examine the variable importance scores resulting from the ensemble

Table 3.6: Final and maximum OOB errors for the wireless indoor localization data.

r	Final OOB error	Minimum OOB error
1	0.101	0.0932
2	0.0294	0.0275
3	0.0175	0.0175
4	0.0156	0.0138
5	0.0138	0.0131
6	0.0144	0.0144
7	0.0156	0.0144

fitting (Table 3.7). From the variable importance table, we see that *WS5*, *WS1*, *WS4* and *WS6* are the most important variables, in decreasing order. The rest of the variables are insignificant, at the $p = 0.10$ significance level.

Table 3.7: Variable importance for the MixForests analysis of the wireless localization data.

Variable	Raw score	z-score	p-value
WS1	227.293	4.588	4.477×10^{-6}
WS2	3.638	0.806	4.200×10^{-1}
WS3	52.586	1.222	2.216×10^{-1}
WS4	160.114	2.431	1.505×10^{-2}
WS5	164.429	8.691	3.608×10^{-18}
WS6	43.140	1.658	9.731×10^{-2}
WS7	26.174	1.419	1.560×10^{-1}

Finally, using our test set of size 400, we will compare mixture discriminant analysis and the mixForest method, using test error as our gauge (Table 3.8). The test error for mixture discriminant analysis is 0.02. If we compare this to our choice of $r = 5$, we see that the mixForest method performs slightly better. For choices $r = 6$ and $r = 7$ we also get better performance for the mixForest method, confirming that there is a sweet spot for choice of r.

Table 3.8: Test errors for the mixForest method.

r	Test error
1	0.0900
2	0.0225
3	0.0250
4	0.0175
5	0.0150
6	0.0150
7	0.0150

For completeness, the confusion matrix for mixForest for $r = 5$ is given in Table 3.9.

Table 3.9: Confusion matrix for $r = 5$, where predicted classes are denoted by overlines.

	$\overline{\text{Zone 1}}$	$\overline{\text{Zone 2}}$	$\overline{\text{Zone 3}}$	$\overline{\text{Zone 4}}$
Zone 1	99	0	1	0
Zone 2	0	96	4	0
Zone 3	0	0	100	0
Zone 4	1	0	0	99

3.6.4 Italian Wine Data

The data consists of 27 chemical and physical observations on 3 types of wine: (1) Barolo, (2) Grignolino, and (3) Barbera. The original data came from Forina *et al.* (1986) and consist of 178 observations. These observations were split into a training set of size 142 and holdout set of size 36. With so many features, a scatter plot of the variable pairs is perhaps of limited use. Therefore, we move onto an analysis of the OOB error convergence for the number of randomly selected features used in ensemble construction. We will focus our attention on the lowest OOB error values.

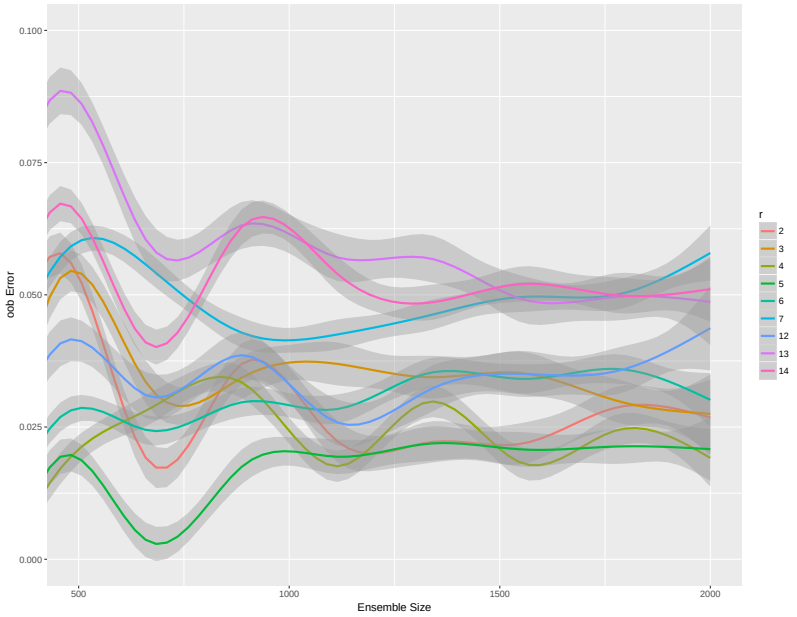

Figure 3.5: OOB error convergence for the Italian wine data.

To make it clearer, we examine an abbreviated table of the lowest OOB errors. There appears to be a sweet spot for choices of r close to 4 or 5. For the rest of this analysis, we will work with $r = 5$.

Table 3.10: Final and minimum OOB error for the wine data for different values of r and $K = 2000$.

r	Final OOB error	Minimum OOB error
2	0.0282	0.0141
3	0.0282	0.0211
4	0.0211	0.00704
5	0.0211	0.00704
6	0.0282	0.0141
7	0.0493	0.0352
12	0.0423	0.0211
13	0.0493	0.0493
14	0.0493	0.0352

To further investigate, we examine the variable importance calculations, but only for the most critical variables (Table 3.11). Beyond these variables, the p-values are greater than 0.10. As can be seen, there are a number of seemingly very significant variables. The most significant include proline, flavanoids, color intensity and alcohol.

Table 3.11: Variable importance for the eight most important variables for the wine data.

Variable	Raw score	z-score	p-value
Proline	13.663	3.191	0.00142
Flavanoids	14.199	3.0425	0.00235
Color intensity	12.170	2.739	0.00616
Alcohol	10.175	2.264	0.0236
OD280/OD315 of diluted wines	9.975	2.166	0.0303
Hue	7.594	1.972	0.0486
OD280/OD315 of flavanoids	7.877	1.793	0.0730
Total phenols	6.813	1.718	0.0858

Finally, we compare the performance of mixture discriminant analysis with the mixForest method using our holdout set of size 36.

Table 3.12: Test errors for the mixForest method.

r	Test error
1	0.0278
2	0.000
3	0.000
4	0.0278
5	0.0278
6	0.0556
7	0.0278
8	0.0278
9	0.0556
10	0.0556

The test error for mixture discriminant analysis is 0.0278. This value is exactly the same as our earlier choice of $r = 5$. However, for $r = 2$ and $r = 3$ there appears to be perfect classification of the holdout set. Below is the confusion matrix for our earlier choice of $r = 5$.

Table 3.13: Confusion matrix for $r = 5$, where predicted classes are overlined.

	$\overline{\text{Barolo}}$	$\overline{\text{Grignolino}}$	$\overline{\text{Barbera}}$
Barolo	12	0	0
Grignolino	0	14	0
Barbera	0	1	9

3.7 Discussion and Summary

3.7.1 Bootstrapping Remarks

There are two possible concerns: (a) the number of random draws d, and (b) the number of bootstrap replicates K. Breiman (1996a) explains that there is no real increase in accuracy for making up to $2N$ random draws on $\mathbf{T}$. However, there is the side affect of decreasing the number of OOB observations for each $\mathbf{T}_{B,k}$. Therefore, the number of random draws is left at N. Other than making each bootstrap replicate larger, it might be possible to increase the number of bootstrap replicates. If this is done, perhaps K can be too large?

Breiman (2001) shows that overfitting is not a concern. As more trees are added to the ensemble a limiting value of the generalization error is produced. In fact, the upper bound for the generalization error can be derived using the strength of each learner and the dependency between learners. This means that there is no concern

of generating too many bootstrap replicates. However, enough bootstrap replicates must be generated to increase the accuracy of our ensemble.

3.7.2 Remarks about Random Feature Selection

Breiman (1996a) shows that bagging provides many advantages, including significant gains in accuracy, under the right conditions. The key to gains in accuracy is the instability of the prediction method used to construct the predictor $F_k(\mathbf{x}, \mathbf{T}_k)$. Meaning, if small changes in the learning set $\mathbf{T}_k$ cause significant changes in the predictor $F_k(\mathbf{x}, \mathbf{T}_k)$, then bagging can improve accuracy. For the most part, this seems like a safe assumption. Breiman *et al.* (1996) show that many popular prediction methods including neural nets, classification and regression trees, subset selection in linear regression are unstable. It is our belief that prediction using Gaussian mixture models will also gain accuracy through bagging.

When referring to accuracy, it is generalization error that is the main concern. Generalization error refers to the error made when predicting outcomes of previously unseen data. This is typically gauged using a cross validation approach, whereby the data is split into training and test sets. In more complicated versions, the data is randomly split into many subsets, and the predictive models are trained and validated many times over. This makes cross validation computationally expensive. However, by bagging we have a built in way of estimating generalization error, which is not only convenient, but which works surprisingly well.

Our primary concern in random feature selection is r, the number of features to randomly select. Larger values of r will yield stronger trees, but stronger correlation

between trees. Because the forest error rate depends on the strength of the trees and the correlation between them, an appropriate value of r must be chosen (Breiman, 2001).

Breiman and Cutler (2004) state that r is the only sensitive parameter in random forests, because it affects the forest error rate. There is an optimal range for r; however, it is wide. Using the OOB error rate (2.3.6), the value of r can be determined empirically.

3.7.3 Remarks Specifically about MixForest

We introduced a new ensemble method called MixForest, with the aim of improving the classification accuracy of MDA. We covered the technical details of using maximum likelihood estimation via an EM algorithm to perform parameter estimation. We also covered using OOB error estimates to fine tune the performance of the analogue ensemble.

Then we demonstrated the accuracy of the analogue ensemble through analysis of three real datasets: the Pima Indians diabetes data, the wireless indoor localization data, and the Italian wine data. We calculated OOB error estimates, variable importance scores, and compared the performance of the analogue ensemble with a Gaussian mixtures and random forests. In so doing, we showed that mixForests perform at least as well as MDA, if not better, while adding inferential advantages. We also showed that the OOB error estimate is a reasonable estimate of the classification accuracy; however, as one might expect, it does not match performance on the test set. In addition, the analogue ensemble allows us to make inferences about

the importance of the variables and to reduce the dimension of the data.

Even so, there is room for improving the ensemble, and while using Gaussian mixtures is adequate for exploring ensemble methods, other types of mixture models can also be employed. In addition to this, there are a number of other possible tweaks that can be made, in order to coax greater classification accuracy from these types of methods. Certainly, the introduction of the random forest analogue for MDA is a take off point for further work, and greater understanding of statistical learning methods.

Chapter 4

MDA for High Dimensional Data

4.1 Motivation

In Chapter 3, a method called the mixForest was proposed with the aim of improving MDA in the context of Gaussian mixture models. The improvements were significant, but not entirely extensible to the case of high-dimensional data. In this chapter, a mixForest is proposed based on using mixtures of factor analyzers. The method will retain all of the benefits of a mixForest, including increases in accuracy, variable importance scores, estimates of generalization error, and dealing with missing data. The new mixForest will also have the added benefit of dimensionality reduction, due to the use of mixtures of factor analyzers. This will allow the mixForest presented in this chapter to handle high dimensional data.

4.2 Factor Analysis Model

As detailed by McNicholas (2016), the factor analysis model is a data reduction method. The goal is to replace p observed variables with q latent factors, such that $q < p$. The more variability the q latent factors explain in the p observed variables, the better the factor analysis model performs. Given N independent p-dimensional random variables $\mathbf{X}_1, \ldots, \mathbf{X}_N$, McNicholas (2016) states the factor analysis model as

$$\mathbf{X}_i = \boldsymbol{\mu} + \boldsymbol{\Lambda}\mathbf{U}_i + \boldsymbol{\epsilon}_i \tag{4.2.1}$$

for $i = 1, \ldots, N$, where $\boldsymbol{\Lambda}$ is a $p \times q$ matrix of factor loadings, the latent factor $\mathbf{U}_i \sim \mathcal{N}(0, \mathbf{I}_q)$, and $\boldsymbol{\epsilon}_i \sim \mathcal{N}(0, \boldsymbol{\Psi})$, where $\boldsymbol{\Psi} = \text{diag}(\psi_1, \ldots, \psi_p)$. Further, both $\mathbf{U}_i$ and $\boldsymbol{\epsilon}_i$ are independently distributed and independent of each other.

Given (4.2.1) and the accompanying constraints, the marginal distribution of $\mathbf{X}_i$ is $\mathcal{N}\left(\boldsymbol{\mu}, \boldsymbol{\Lambda}\boldsymbol{\Lambda}^\top + \boldsymbol{\Psi}\right)$. Further, the covariance matrix $\boldsymbol{\Lambda}\boldsymbol{\Lambda}^\top + \boldsymbol{\Psi}$ has

$$pq + p - \frac{1}{2}q(q-1)$$

free parameters (Lawley and Maxwell, 1962). The reduction of free parameters in the covariance structure, using the factor analysis model, will be

$$\frac{1}{2}p(p+1) - \left[pq + p - \frac{1}{2}q(q-1)\right] = \frac{1}{2}\left[(p-q)^2 - (p+q)\right].$$

Meaning, the following constraint must hold for the factor analysis model to result

in dimension reduction:

$$(p-q)^2 > (p+q)$$

(Lawley and Maxwell, 1962).

4.2.1 Parameter Estimation for the Factor Analysis Model

A complete treatment of parameter estimation for the factor analysis model is found in McNicholas (2016). Our goal is to obtain estimates for $\boldsymbol{\mu}$, $\boldsymbol{\Lambda}$, and $\boldsymbol{\Psi}$. It is straightforward to estimate $\boldsymbol{\mu}$ by using maximum likelihood estimation, but for $\boldsymbol{\Lambda}$ and $\boldsymbol{\Psi}$ an EM algorithm must be used.

To begin, the log-likelihood of the factor analysis model, given observed data $\mathbf{x}_1, \ldots, \mathbf{x}_N$, is

$$\begin{aligned} \ell(\boldsymbol{\mu}, \boldsymbol{\Lambda}, \boldsymbol{\Psi}) &= \sum_{i=1}^{N} \log \phi\left(\mathbf{x}_i \mid \boldsymbol{\mu}, \boldsymbol{\Lambda}\boldsymbol{\Lambda}^\top + \boldsymbol{\Psi}\right), \\ &= -\frac{Np}{2}\log 2\pi - \frac{N}{2}\log\left|\boldsymbol{\Lambda}\boldsymbol{\Lambda}^\top + \boldsymbol{\Psi}\right| - \frac{N}{2}\operatorname{tr}\left\{\mathbf{S}\left(\boldsymbol{\Lambda}\boldsymbol{\Lambda}^\top + \boldsymbol{\Psi}\right)^{-1}\right\}, \end{aligned} \tag{4.2.2}$$

where

$$\mathbf{S} = \frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{x}_i - \boldsymbol{\mu}\right)\left(\mathbf{x}_i - \boldsymbol{\mu}\right)^\top.$$

Differentiating (4.2.2) with respect to $\boldsymbol{\mu}$ and setting the resulting function equal to zero, yields

$$\hat{\boldsymbol{\mu}} = \overline{\mathbf{x}}.$$

To get the estimates for $\boldsymbol{\Lambda}$ and $\boldsymbol{\Psi}$ the EM algorithm is employed beginning with the E-step.

E-step

As summarized in Section 3.3.1, the E-step is the calculation of the expected value of the complete-data log-likelihood conditioned on the parameter estimates from the previous step. To construct the complete-data log-likelihood we first take into account the conditional distribution: $\mathbf{X}_i \mid \mathbf{u}_i \sim \mathcal{N}(\boldsymbol{\mu} + \boldsymbol{\Lambda}\mathbf{u}_i, \boldsymbol{\Psi})$, giving the following log-likelihood:

$$\begin{aligned}\log \mathrm{f}(\mathbf{x}_i \mid \mathbf{u}_i) =& -\frac{p}{2}\log 2\pi - \frac{1}{2}\log|\boldsymbol{\Psi}| - \frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu} - \boldsymbol{\Lambda}\mathbf{u}_i)^\top \boldsymbol{\Psi}^{-1}(\mathbf{x}_i - \boldsymbol{\mu} - \boldsymbol{\Lambda}\mathbf{u}_i),\\ =& -\frac{p}{2}\log 2\pi - \frac{1}{2}\log|\boldsymbol{\Psi}| - \frac{1}{2}\operatorname{tr}\left\{\boldsymbol{\Psi}^{-1}(\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^\top\right\}\\ &+ (\mathbf{x}_i - \boldsymbol{\mu})^\top \boldsymbol{\Psi}^{-1}\boldsymbol{\Lambda}\mathbf{u}_i - \frac{1}{2}\operatorname{tr}\left\{\boldsymbol{\Lambda}^\top\boldsymbol{\Psi}^{-1}\boldsymbol{\Lambda}\mathbf{u}_i\mathbf{u}_i^\top\right\}.\end{aligned}$$

Secondly, we take into account the complete-data: (1) the observed data $\mathbf{x}_1, \ldots, \mathbf{x}_N$ and (2) the latent factors $\mathbf{u}_i, \ldots, \mathbf{u}_N$. Note that the observed data is formed of p-dimensional vectors $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})'$ and the latent factors are formed of q-dimensional vectors $\mathbf{u}_i = (u_{i1}, \ldots, u_{iq})'$. This yields the following complete-data log-likelihood:

$$\begin{aligned}\ell(\boldsymbol{\mu}, \boldsymbol{\Lambda}, \boldsymbol{\Psi}) =& \sum_{i=1}^{N} \log\left[\mathrm{f}(\mathbf{x}_i \mid \mathbf{u}_i)\,\mathrm{f}(\mathbf{u}_i)\right]\\ =& C - \frac{N}{2}\log|\boldsymbol{\Psi}| - \frac{1}{2}\operatorname{tr}\left\{\boldsymbol{\Psi}^{-1}\sum_{i=1}^{N}(\mathbf{x}_i - \mathbf{u}_i)(\mathbf{x}_i - \mathbf{u}_i)^\top\right\}\\ &+ \sum_{i=1}^{N}(\mathbf{x}_i - \mathbf{u}_i)^\top\boldsymbol{\Psi}^{-1}\boldsymbol{\Lambda}\mathbf{u}_i - \frac{1}{2}\operatorname{tr}\left\{\boldsymbol{\Lambda}^\top\boldsymbol{\Psi}^{-1}\boldsymbol{\Lambda}\sum_{i=1}^{N}\mathbf{u}_i\mathbf{u}_i^\top\right\},\end{aligned}$$

where C is constant with respect to $\boldsymbol{\mu}$, $\boldsymbol{\Lambda}$, and $\boldsymbol{\Psi}$.

Following the approach and using the notation of McNicholas (2016), parameter estimation of the factor analysis model continues with the calculation of $\mathcal{Q}(\boldsymbol{\Lambda}, \boldsymbol{\Psi})$, which is the expected value of $\ell(\boldsymbol{\mu}, \boldsymbol{\Lambda}, \boldsymbol{\Psi})$.

$$\begin{aligned}\mathcal{Q}(\boldsymbol{\Lambda}, \boldsymbol{\Psi}) =& \mathrm{C} - \frac{N}{2}\log|\boldsymbol{\Psi}| - \frac{1}{2}\operatorname{tr}\left\{\boldsymbol{\Psi}^{-1}\sum_{i=1}^{N}(\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^{\top}\right\} \\ &+ \sum_{i=1}^{N}(\mathbf{x}_i - \overline{\mathbf{x}})^{\top}\boldsymbol{\Psi}^{-1}\boldsymbol{\Lambda}\mathbb{E}[\mathbf{U}_i \mid \mathbf{x}_i] - \frac{1}{2}\operatorname{tr}\left\{\boldsymbol{\Lambda}^{\top}\boldsymbol{\Psi}^{-1}\boldsymbol{\Lambda}\sum_{i=1}^{N}\mathbb{E}\left[\mathbf{U}_i\mathbf{U}_i^{\top} \mid \mathbf{x}_i\right]\right\}.\end{aligned}$$

To continue the calculation of $\mathcal{Q}(\boldsymbol{\Lambda}, \boldsymbol{\Psi})$, both $\mathbb{E}[\mathbf{U}_i \mid \mathbf{x}_i]$ and $\mathbb{E}\left[\mathbf{U}_i\mathbf{U}_i^{\top} \mid \mathbf{x}_i\right]$ must be resolved using the joint distribution of $\mathbf{X}_i$ and $\mathbf{U}_i$:

$$\begin{bmatrix}\mathbf{X}_i \\ \mathbf{U}_i\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu} \\ \mathbf{0}\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Lambda}\boldsymbol{\Lambda}^{\top} + \boldsymbol{\Psi} & \boldsymbol{\Lambda} \\ \boldsymbol{\Lambda}^{\top} & \mathbf{I}_q\end{bmatrix}\right).$$

Then the conditional expectations are

$$\mathbb{E}[\mathbf{U}_i \mid \mathbf{x}_i] = \boldsymbol{\beta}(\mathbf{x}_i - \boldsymbol{\mu}).$$

where $\boldsymbol{\beta} = \boldsymbol{\Lambda}^{\top}\left(\boldsymbol{\Lambda}\boldsymbol{\Lambda}^{\top} + \boldsymbol{\Psi}\right)^{-1}$, and

$$\begin{aligned}\mathbb{E}\left[\mathbf{U}_i\mathbf{U}_i^{\top} \mid \mathbf{x}_i\right] =& \mathbb{V}\mathrm{ar}[\mathbf{U}_i \mid \mathbf{x}_i] + \mathbb{E}[\mathbf{U}_i \mid \mathbf{x}_i]\,\mathbb{E}[\mathbf{U}_i \mid \mathbf{x}_i]^{\top} \\ =& \mathbf{I}_q - \boldsymbol{\beta}\boldsymbol{\Lambda} + \boldsymbol{\beta}(\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^{\top}\boldsymbol{\beta}^{\top}.\end{aligned}$$

Finally, using these expected values yields:

$$\mathcal{Q}\left(\mathbf{\Lambda}, \mathbf{\Psi}\right) = \mathrm{C} + \frac{N}{2}\log\left|\mathbf{\Psi}^{-1}\right| - \frac{N}{2}\operatorname{tr}\left\{\mathbf{\Psi}^{-1}\mathbf{S}_{\overline{\mathbf{x}}}\right\} + N\operatorname{tr}\left\{\mathbf{\Psi}^{-1}\hat{\boldsymbol{\beta}}\mathbf{S}_{\overline{\mathbf{x}}}\right\} - \frac{N}{2}\operatorname{tr}\left\{\mathbf{\Lambda}^{\top}\mathbf{\Psi}^{-1}\mathbf{\Lambda}^{\mathbf{\Theta}}\right\},$$

where $\mathbf{\Theta} = \mathbf{I}_q - \hat{\boldsymbol{\beta}}\hat{\mathbf{\Lambda}} + \hat{\boldsymbol{\beta}}\mathbf{S}_{\overline{\mathbf{x}}}\hat{\boldsymbol{\beta}}^{\top}$ is a symmetric $q \times q$ matrix, $\hat{\boldsymbol{\beta}} = \hat{\mathbf{\Lambda}}^{\top}\left(\hat{\mathbf{\Lambda}}\hat{\mathbf{\Lambda}}^{\top} + \hat{\mathbf{\Psi}}\right)^{-1}$, and

$$\mathbf{S}_{\overline{\mathbf{x}}} = \frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{x}_i - \overline{\mathbf{x}}\right)\left(\mathbf{x}_i - \overline{\mathbf{x}}\right)^{\top}.$$

M-step

To get parameter estimates for $\mathbf{\Lambda}$ and $\mathbf{\Psi}$ the expected value of the complete-data log-likelihood is differentiated and solved in the usual manner:

$$\begin{aligned}
\mathcal{S}_1 =& \frac{\partial \mathcal{Q}}{\partial \mathbf{\Lambda}} \\
=& N\frac{\partial}{\partial \mathbf{\Lambda}}\operatorname{tr}\left\{\mathbf{\Psi}^{-1}\mathbf{\Lambda}\hat{\boldsymbol{\beta}}\mathbf{S}_{\overline{\mathbf{x}}}\right\} - \frac{N}{2}\frac{\partial}{\partial \mathbf{\Lambda}}\operatorname{tr}\left\{\mathbf{\Lambda}^{\top}\mathbf{\Psi}^{-1}\mathbf{\Lambda}\mathbf{\Theta}\right\} \\
=& N\left(\mathbf{\Psi}^{-1}\right)^{\top}\left(\hat{\boldsymbol{\beta}}\mathbf{S}_{\overline{\mathbf{x}}}\right)^{\top} - \frac{N}{2}\frac{\partial}{\partial \mathbf{\Lambda}}\operatorname{tr}\left\{\mathbf{\Lambda}\mathbf{\Theta}\mathbf{\Lambda}^{\top}\mathbf{\Psi}^{-1}\right\} \\
=& N\mathbf{\Psi}^{-1}\mathbf{S}_{\overline{\mathbf{x}}}^{\top}\hat{\boldsymbol{\beta}}^{\top} - \frac{N}{2}\left[\left(\mathbf{\Psi}^{-1}\right)^{\top}\mathbf{\Lambda}\mathbf{\Theta}^{\top} + \mathbf{\Psi}^{-1}\mathbf{\Lambda}\mathbf{\Theta}\right] \\
=& N\mathbf{\Psi}^{-1}\mathbf{S}_{\overline{\mathbf{x}}}\hat{\boldsymbol{\beta}}^{\top} - N\mathbf{\Psi}^{-1}\mathbf{\Lambda}\mathbf{\Theta},
\end{aligned}$$

and

$$\begin{aligned}
\mathcal{S}_2\left(\mathbf{\Lambda}, \mathbf{\Psi}\right) =& \frac{\partial \mathcal{Q}}{\partial \mathbf{\Psi}^{-1}} \\
=& \frac{N}{2}\mathbf{\Psi} - \frac{N}{2}\mathbf{S}_{\overline{\mathbf{x}}}^{\top} + N\left(\mathbf{\Lambda}\hat{\boldsymbol{\beta}}\mathbf{S}_{\overline{\mathbf{x}}}\right)^{\top} - \frac{N}{2}\left(\mathbf{\Lambda}^{\top}\right)^{\top}\left(\mathbf{\Lambda}\mathbf{\Theta}\right)^{\top} \\
=& \frac{N}{2}\mathbf{\Psi} - \frac{N}{2}\mathbf{S}_{\overline{\mathbf{x}}}^{\top} + N\mathbf{\Lambda}\hat{\boldsymbol{\beta}}\mathbf{S}_{\overline{\mathbf{x}}} - \frac{N}{2}\mathbf{\Lambda}\mathbf{\Theta}^{\top}\mathbf{\Lambda}^{\top}.
\end{aligned}$$

Setting $\mathcal{S}_1\left(\hat{\mathbf{\Lambda}}^{\text{new}}, \hat{\mathbf{\Psi}}^{\text{new}}\right) = 0$ and $\text{diag}\left\{\mathcal{S}_2\left(\hat{\mathbf{\Lambda}}^{\text{new}}, \hat{\mathbf{\Psi}}^{\text{new}}\right)\right\} = 0$ and solving, yields

$$\hat{\mathbf{\Lambda}}^{\text{new}} = \mathbf{S}_{\overline{\mathbf{x}}}\hat{\boldsymbol{\beta}}^{\top}\mathbf{\Theta}^{-1},$$

$$\begin{aligned}\hat{\mathbf{\Psi}}^{\text{new}} =& \text{diag}\left\{\mathbf{S}_{\overline{\mathbf{x}}}^{\top} - 2\hat{\mathbf{\Lambda}}^{\text{new}}\hat{\boldsymbol{\beta}}\mathbf{S}_{\overline{\mathbf{x}}} + \hat{\mathbf{\Lambda}}^{\text{new}}\mathbf{\Theta}^{\top}\left(\mathbf{\Theta}^{-1}\right)^{\top}\hat{\boldsymbol{\beta}}\mathbf{S}_{\overline{\mathbf{x}}}\right\} \\ =& \text{diag}\left\{\mathbf{S}_{\overline{\mathbf{x}}} - \hat{\mathbf{\Lambda}}^{\text{new}}\hat{\boldsymbol{\beta}}\mathbf{S}_{\overline{\mathbf{x}}}\right\}.\end{aligned}$$

The EM Algorithm as given in McNicholas (2016), alternates between the E-step and the M-step, until convergence is attained, and can be phrased in the following manner.

Algorithm 1 EM Algorithm for Factor Analysis Model

compute $\mathbf{S}_{\overline{\mathbf{x}}}$
initalize $\mathbf{\Lambda}$ and $\mathbf{\Psi}$
while convergence criterion not met **do**
  compute $\hat{\boldsymbol{\beta}} = \hat{\mathbf{\Lambda}}^{\top} = \left(\hat{\mathbf{\Lambda}}\hat{\mathbf{\Lambda}}^{\top} + \hat{\mathbf{\Psi}}\right)^{-1}$
  compute update $\mathbf{\Theta} = \mathbf{I}_q - \hat{\boldsymbol{\beta}}\hat{\mathbf{\Lambda}} + \hat{\boldsymbol{\beta}}\mathbf{S}_{\overline{\mathbf{x}}}\hat{\boldsymbol{\beta}}^{\top}$
  update $\hat{\mathbf{\Lambda}}^{\text{new}} = \mathbf{S}_{\overline{\mathbf{x}}}\hat{\boldsymbol{\beta}}^{\top}\mathbf{\Theta}^{-1}$
  update $\hat{\mathbf{\Psi}}^{\text{new}} = \text{diag}\left\{\mathbf{S}_{\overline{\mathbf{x}}} - \hat{\mathbf{\Lambda}}^{\text{new}}\hat{\boldsymbol{\beta}}\mathbf{S}_{\overline{\mathbf{x}}}\right\}$
  check convergence criterion $\hat{\mathbf{\Lambda}} \leftarrow \hat{\mathbf{\Lambda}}^{\text{new}}$, $\hat{\mathbf{\Psi}} \leftarrow \hat{\mathbf{\Psi}}^{\text{new}}$
end while

There are two considerations when employing an EM Algorithm: starting and stopping.

Starting the EM Algorithm

For the purposes of this paper, the `pgmm` package (McNicholas *et al.*, 2018) is used. To initialize the parameters in the EM algorithm three methods are available: random,

k-means, and user-specified. For the last option of a user-specified starting point any form of clustering, including hierarchical clustering, can be employed to get the starting points.

Stopping Criterion for the EM Algorithm

As described by McLachlan and Peel (2000a) and McNicholas (2016), the EM algorithm is usually stopped when from one iteration to the next, the parameter estimates or the log-likelihood exhibit very little change. When phrased in terms of the log-likelihood the EM algorithm is stopped when

$$l^{(k)} = \log L(\mathbf{v}^{(k)}),$$

the log-likelihood at the kth step is only a little smaller than the log-likelihood at the $k+1$th step by a predetermined threshold, i.e.,

$$l^{(k+1)} - l^{(k)} < \epsilon$$

for ϵ small. But, this only captures the lack of progress made by the algorithm. When convergence is not smooth, this might lead to prematurely stopping the EM algorithm, so we can also consider criteria based on Aitken's acceleration:

$$a^{(k)} = \frac{l^{(k+1)} - l^{(k)}}{l^{(k)} - l^{(k-1)}},$$

which is the acceleration at the kth step. If it is assumed that the log-likelihood will eventually converge (which is reasonable), then the asymptotic log-likelihood at the $k+1$th step is

$$l_{\infty}^{(k+1)} = l^{(k)} + \frac{l^{(k+1)} - l^{(k)}}{1 - a^{(k)}}$$

and given the difference is positive a preferred stopping criterion is

$$l_{\infty}^{(k+1)} - l^{(k)} < \epsilon$$

for ϵ small (McNicholas *et al.*, 2010). For alternative stopping rules see McNicholas (2016).

The Woodbury Identity

During the E-step of the EM algorithm for the factor analysis model, the estimate $\hat{\boldsymbol{\beta}} = \hat{\boldsymbol{\Lambda}}^{\top}\left(\hat{\boldsymbol{\Lambda}}\hat{\boldsymbol{\Lambda}}^{\top} + \hat{\boldsymbol{\Psi}}\right)^{-1}$ requires the inversion of a $p \times p$ matrix. Large values of p make this inversion computationally expensive (McNicholas, 2016).

The Woodbury identity introduced in Woodbury (1950), allows us to avoid these expensive matrix inversions. Given matrices $\mathbf{A}$ $(m \times m)$, $\mathbf{U}$ $(m \times k)$, $\mathbf{C}$ $(k \times k)$, and $\mathbf{V}$ $(k \times m)$, the Woodbury identity is

$$(\mathbf{A} + \mathbf{UCV})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}\left(\mathbf{C}^{-1} + \mathbf{VA}^{-1}\mathbf{U}\right)^{-1}\mathbf{VA}^{-1}.$$

In the case of factor analysis, set $\mathbf{U} = \boldsymbol{\Lambda}$, $\mathbf{V} = \boldsymbol{\Lambda}^{\top}$, $\mathbf{A} = \boldsymbol{\Psi}$, and $\mathbf{C} = \mathbf{I}_q$, so that the

Woodbury identity becomes

$$\left(\boldsymbol{\Psi}+\boldsymbol{\Lambda}\boldsymbol{\Lambda}^{\top}\right)^{-1}=\boldsymbol{\Psi}^{-1}-\boldsymbol{\Psi}^{-1}\boldsymbol{\Lambda}\left(\mathbf{I}_{q}\boldsymbol{\Lambda}^{\top}\boldsymbol{\Psi}\boldsymbol{\Lambda}\right)^{-1}\boldsymbol{\Lambda}^{\top}\boldsymbol{\Psi}^{-1}. \tag{4.2.3}$$

The right side of (4.2.3) requires the inversion of a $p\times p$ matrix, while the left side only requires the inversion of a $q\times q$ matrix. If the factor analysis model performs well, i.e., if $q\ll p$, this identity will save significant amounts of computation time. The related identity for the determinant

$$\left|\boldsymbol{\Lambda}\boldsymbol{\Lambda}^{\top}+\boldsymbol{\Psi}\right|=\frac{|\boldsymbol{\Psi}|}{\left|\mathbf{I}_{q}-\boldsymbol{\Lambda}^{\top}\left(\boldsymbol{\Lambda}\boldsymbol{\Lambda}^{\top}+\boldsymbol{\Psi}\right)^{-1}\boldsymbol{\Lambda}\right|}.$$

allows further time saving (McLachlan and Peel, 2000a).

4.3 Mixture of Factor Analyzers

As detailed in McNicholas (2016), for the mixture of factor analyzers model, we assume that each $\mathbf{X}_i$ can be written as

$$\mathbf{X}_i=\boldsymbol{\mu}_g+\boldsymbol{\Lambda}_g\mathbf{U}_{ig}+\boldsymbol{\epsilon}_{ig}$$

with a probability of π_g, for $i=1,\ldots,N$ and $g=1,\ldots,G$ where $\sum_{g=1}^{G}\pi_g=1$, $\boldsymbol{\Lambda}_g$ is matrix of factor loadings of dimensions $p\times q$, $\mathbf{U}_{ig}$ are independent $\mathcal{N}\left(0,\mathbf{I}_q\right)$, $\boldsymbol{\epsilon}_{ig}$ are independent $\mathcal{N}\left(\mathbf{0},\boldsymbol{\Psi}_g\right)$, where $\boldsymbol{\Psi}_g$ is a $p\times p$ diagonal matrix with positive diagonal elements, and $\mathbf{U}_{ig}$ are independent of $\boldsymbol{\epsilon}_{ig}$.

If the above conditions are true, then each observation $\mathbf{x}_i$ is represented by the following mixture of G densities with mixing proportions π_g $(g = 1, \ldots, G)$:

$$\mathrm{f}\left(\mathbf{x}_i \mid \boldsymbol{\vartheta}\right) = \sum_{g=1}^{G} \pi_g \phi\left(\mathbf{x}_i \mid \boldsymbol{\mu}_g, \boldsymbol{\Lambda}_g \boldsymbol{\Lambda}_g^{\top} + \boldsymbol{\Psi}_g\right),$$

where $\boldsymbol{\vartheta}$ is a vector of model parameters consisting of $\boldsymbol{\mu}_g$, $\boldsymbol{\Lambda}_g$, $\boldsymbol{\Psi}_g$, and π_g for $g = 1, \ldots, G$.

McLachlan and Peel (2000b) further details that when this mixture of factor analyzers is adopted it gives a local dimensionality reduction. When fitting a mixture of factor analyzers with normal components, placing no restriction on the parameters, each $\boldsymbol{\Sigma}_g = \boldsymbol{\Lambda}_g \boldsymbol{\Lambda}_g^{\top} + \boldsymbol{\Psi}_g$ has $\frac{1}{2}p(p+1)$ free parameters. When the number of components (G) grows large compared to the sample size (N), overfitting becomes an issue. Since each $\boldsymbol{\Lambda}_g$ has dimensions $p \times q$, when $q << p$ there are savings in computation time.

4.3.1 Parsimonious Gaussian Mixture Models

From Section 4.3 we see that the covariance structure is represented by $\boldsymbol{\Sigma}_g = \boldsymbol{\Lambda}_g \boldsymbol{\Lambda}_g^{\top} + \boldsymbol{\Psi}_g$. By placing constraints on $\boldsymbol{\Sigma}_g$ it's possible to create models with a variety of behaviours. In addition to the local dimensional reduction inherent in the mixture of factor analyzers model, placing constraints on $\boldsymbol{\Sigma}_g$ creates parsimony and this will lead to an efficient use of resources.

McNicholas and Murphy (2008) details three specific constraints: (1) $\boldsymbol{\Lambda}_g = \boldsymbol{\Lambda}$, (2) $\boldsymbol{\Psi}_g = \boldsymbol{\Psi}$, and (3) $\boldsymbol{\Psi}_g = \psi_g \mathbf{I}_p$. Each combination of constraints produces one

of eight models, with varying numbers of free parameters, summarized in Table 4.1. These eight models are called the family of PGMMs.

Table 4.1: The covariance structure of the PGMM family of models, where 'C' denotes constrained and 'U' denotes unconstrained.

$\boldsymbol{\Lambda}_g = \boldsymbol{\Lambda}$	$\boldsymbol{\Psi}_g = \boldsymbol{\Psi}$	$\boldsymbol{\Psi}_g = \boldsymbol{\psi}_g \mathbf{I}_p$	$\boldsymbol{\Sigma}_g$	Free Cov. Parameters
C	C	C	$\boldsymbol{\Lambda}\boldsymbol{\Lambda}^\top + \boldsymbol{\psi}\mathbf{I}_p$	$pq - q(q-1)/2 + 1$
C	C	U	$\boldsymbol{\Lambda}\boldsymbol{\Lambda}^\top + \boldsymbol{\Psi}$	$pq - q(q-1)/2 + p$
C	U	C	$\boldsymbol{\Lambda}\boldsymbol{\Lambda}^\top + \boldsymbol{\psi}_g\mathbf{I}_p$	$pq - q(q-1)/2 + G$
C	U	U	$\boldsymbol{\Lambda}\boldsymbol{\Lambda}^\top + \boldsymbol{\Psi}_g$	$pq - q(q-1)/2 + Gp$
U	C	C	$\boldsymbol{\Lambda}_g\boldsymbol{\Lambda}_g^\top + \boldsymbol{\psi}\mathbf{I}_p$	$G\left[pq - q\left(q-1\right)/2\right] + 1$
U	C	U	$\boldsymbol{\Lambda}_g\boldsymbol{\Lambda}_g^\top + \boldsymbol{\Psi}$	$G\left[pq - q\left(q-1\right)/2\right] + p$
U	U	C	$\boldsymbol{\Lambda}_g\boldsymbol{\Lambda}_g^\top + \boldsymbol{\psi}_g\mathbf{I}_p$	$G\left[pq - q\left(q-1\right)/2\right] + G$
U	U	U	$\boldsymbol{\Lambda}_g\boldsymbol{\Lambda}_g^\top + \boldsymbol{\Psi}_g$	$G\left[pq - q\left(q-1\right)/2\right] + Gp$

The usefulness of this family of models has been demonstrated previously in model-based classifcation (McNicholas, 2010), and model-based discriminant analysis (Andrews and McNicholas, 2011).

Parameter Estimation for the PGMM Family

The EM algorithm is useful in the context of estimation problems when missing data are present. In the case of mixtures of factor analyzers, there are two sources of missing data: the component memberships z_{ig}, and the latent factors $\mathbf{u}_{ig}$, for $i = 1, \ldots, N$, $g = 1, \ldots, G$. Thus, a variant of the EM algorithm must be used. The alternating expectation-conditional maximization (AECM) algorithm was proposed by Meng and Van Dyk (1997), and is an extension of the expectation-conditional

maximization algorithm introduced by Meng and Rubin (1993). The AECM algorithm allows for alternation between two stages, where each source of missing data can be addressed. Details for this procedure are laid out by McNicholas (2016), and for the more general case details can be found in McLachlan and Peel (2000b).

AECM: Stage 1

Adhering to the notation and methodology used in McNicholas (2016), for the first stage of the AECM the complete data are comprised of the observed feature vectors $(\mathbf{x}_1, \ldots, \mathbf{x}_N)$ and the unobserved component membership vectors $(\mathbf{z}_1, \ldots, \mathbf{z}_N)$, as defined in Section 3.2. In this stage the expected value of the complete-data log-likelihood will be used to estimate π_g and $\boldsymbol{\mu}_g$ for $g = 1, \ldots, G$. The likelihood equation for this stage is

$$\ell_1 = \sum_{i=1}^{N} \sum_{g=1}^{G} z_{ig} \log \left[\pi_g \phi \left(\mathbf{x}_i \mid \boldsymbol{\mu}_g, \boldsymbol{\Lambda}_g \boldsymbol{\Lambda}_g^\top + \boldsymbol{\Psi}_g \right) \right]. \tag{4.3.1}$$

The conditional expected values of the component membership labels are

$$\hat{z}_{ig} = \frac{\hat{\pi}_g \phi \left(\mathbf{x}_i \mid \hat{\boldsymbol{\mu}}_g, \hat{\boldsymbol{\Lambda}}_g \hat{\boldsymbol{\Lambda}}_g^\top + \hat{\boldsymbol{\Psi}}_g \right)}{\sum_{h=1}^{G} \hat{\pi}_h \phi \left(\mathbf{x}_i \mid \hat{\boldsymbol{\Lambda}}_h \hat{\boldsymbol{\Lambda}}_h^\top + \hat{\boldsymbol{\Psi}}_h \right)}, \tag{4.3.2}$$

for $i = 1, \ldots, N$ and $g = 1, \ldots, G$. Using (4.3.1) and (4.3.13), the expected value of the complete-data log-likelihood for stage 1 of the AECM is

$$\begin{aligned}\mathcal{Q}_1 &= \sum_{i=1}^{N}\sum_{g=1}^{G} \hat{z}_{ig}\left[\log \pi_g + \log \phi\left(\mathbf{x}_i \mid \boldsymbol{\mu}_g, \boldsymbol{\Lambda}_g\boldsymbol{\Lambda}_g^\top + \boldsymbol{\Psi}_g\right)\right] \\ &= \sum_{g=1}^{G} N_g \log \pi_g - \frac{Np}{2}\log 2\pi - \sum_{g=1}^{G}\frac{N_g}{2}\log\left|\boldsymbol{\Lambda}_g\boldsymbol{\Lambda}_g^\top + \boldsymbol{\Psi}_g\right| \\ &\quad - \sum_{g=1}^{G}\frac{N_g}{2}\operatorname{tr}\left\{\mathbf{S}_g\left(\boldsymbol{\Lambda}_g\boldsymbol{\Lambda}_g^\top + \boldsymbol{\Psi}_g\right)^{-1}\right\},\end{aligned}$$

where $N_g = \sum_{g=1}^{G} \hat{z}_{ig}$ and

$$\mathbf{S}_g = \frac{1}{N_g}\sum_{i=1}^{N} \hat{z}_{ig}\left(\mathbf{x}_i - \boldsymbol{\mu}_g\right)\left(\mathbf{x}_i - \boldsymbol{\mu}_g\right)^\top. \tag{4.3.3}$$

Maximizing $\mathcal{Q}_1$ w.r.t. π_g gives

$$\hat{\pi}_g = \frac{N_g}{N} \tag{4.3.4}$$

and maximizing $\mathcal{Q}_1$ w.r.t. $\boldsymbol{\mu}_g$ gives

$$\hat{\boldsymbol{\mu}}_g = \frac{\sum_{i=1}^{N}\hat{z}_{ig}\mathbf{x}_i}{\sum_{i=1}^{N}\hat{z}_{ig}}. \tag{4.3.5}$$

AECM: Stage 2

Following the methods used in McNicholas (2016), the complete-data log-likelihood for the second stage of the AECM algorithm is constructed by including the observed feature vectors $(\mathbf{x}_1, \ldots, \mathbf{x}_N)$, the unobserved component membership vectors $(\mathbf{z}_1, \ldots, \mathbf{z}_N)$, and the latent factors $(\mathbf{u}_{ig})$ for $i = 1, \ldots, N$ and $g = 1, \ldots, G$. The

parameters $\boldsymbol{\Lambda}_g$ and $\boldsymbol{\Psi}_g$ are estimated for $g = 1, \ldots, G$. Since the PGMM family inherits the covariance structure in the factor analysis model, stage 2 of the AECM is analogous to the EM algorithm shown in 4.2.1. The complete-data log-likelihood for stage 2 is

$$\begin{aligned}
\ell_2 &= \sum_{i=1}^{N}\sum_{g=1}^{G} \hat{z}_{ig} \left[\log \pi_g + \log \mathrm{f}\left(\mathbf{x}_i \mid \mathbf{u}_i\right) + \log \mathrm{f}\left(\mathbf{u}_i\right)\right] \\
&= C + \sum_{g=1}^{G}\left[-\frac{N_g}{2}\log\left|\boldsymbol{\Psi}_g\right| - \frac{N_g}{2}\operatorname{tr}\left\{\boldsymbol{\Psi}_g^{-1}\mathbf{S}_g + \sum_{i=1}^{N} z_{ig}\left(\mathbf{x}_i - \boldsymbol{\mu}_g\right)^{\top}\boldsymbol{\Psi}_g^{-1}\boldsymbol{\Lambda}_g\mathbf{u}_i\right\}\right] \\
&\quad - \sum_{g=1}^{G}\left[\frac{1}{2}\operatorname{tr}\left\{\boldsymbol{\Lambda}_g^{\top}\boldsymbol{\Psi}_g^{-1}\boldsymbol{\Lambda}_g\sum_{i=1}^{N} z_{ig}\mathbf{u}_i\mathbf{u}_i^{\top}\right\}\right],
\end{aligned}$$

where C is constant with respect to the parameters $\boldsymbol{\Lambda}_g$ and $\boldsymbol{\Psi}_g$. After conditioning on the parameter estimates, the expected value of the complete-data log-likelihood is

$$\begin{aligned}
\mathcal{Q}_2 =& C + \sum_{g=1}^{G}\left[-\frac{N_g}{2}\log\left|\boldsymbol{\Psi}_g\right| - \frac{N_g}{2}\operatorname{tr}\left\{\boldsymbol{\Psi}_g^{-1}\mathbf{S}_g\right\}\right. \\
&+ \sum_{i=1}^{N}\hat{z}_{ig}\left(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_g\right)^{\top}\boldsymbol{\Psi}_g^{-1}\boldsymbol{\Lambda}_g\mathbb{E}\left[\mathbf{U}_{ig} \mid \mathbf{x}_i, z_{ig} = 1\right] \\
&\left. - \frac{1}{2}\operatorname{tr}\left\{\boldsymbol{\Lambda}_g^{\top}\boldsymbol{\Psi}_g^{-1}\boldsymbol{\Lambda}_g\sum_{i=1}^{N}\hat{z}_{ig}\mathbb{E}\left[\mathbf{U}_{ig}\mathbf{U}_{ig}^{\top} \mid \mathbf{x}_i, z_{ig} = 1\right]\right\}\right] \\
=& C + \frac{1}{2}\sum_{g=1}^{G} N_g\left[\log\left|\boldsymbol{\Psi}^{-1}\right| - \operatorname{tr}\left\{\boldsymbol{\Psi}^{-1}\mathbf{S}_g\right\} + 2\operatorname{tr}\left\{\boldsymbol{\Psi}_g^{-1}\boldsymbol{\Lambda}_g\hat{\boldsymbol{\beta}}_g\mathbf{S}_g\right\}\right. \\
&\left. - \operatorname{tr}\left\{\boldsymbol{\Lambda}_g^{\top}\boldsymbol{\Psi}_g^{-1}\boldsymbol{\Lambda}_g\boldsymbol{\Theta}_g\right\}\right],
\end{aligned}$$

where $\hat{\boldsymbol{\beta}}_g = \boldsymbol{\Lambda}_g^\top \left(\boldsymbol{\Lambda}_g \boldsymbol{\Lambda}_g^\top + \hat{\boldsymbol{\Psi}}_g\right)^{-1}$ and $\boldsymbol{\Theta}_g = \mathbf{I}_q - \hat{\boldsymbol{\beta}}_g \hat{\boldsymbol{\Lambda}}_g + \hat{\boldsymbol{\beta}}_g \mathbf{S}_g \hat{\boldsymbol{\beta}}_g^\top$. At this point, all the parameters have been replaced by their estimates.

The next part of the second stage of the AECM depends on the particular constraints of the model being estimated (refer to Table 4.1). McNicholas (2016) details the remainder of the AECM algorithm for model 'CCU'. For this model, $\boldsymbol{\Lambda}_g = \boldsymbol{\Lambda}$ and $\boldsymbol{\Psi}_g = \boldsymbol{\Psi}$. Using these constraints the expected value of the complete-data log-likelihood becomes

$$\mathcal{Q}_2\left(\boldsymbol{\Lambda}, \boldsymbol{\Psi}\right) = C + \frac{N}{2}\left[\log\left|\boldsymbol{\Psi}^{-1}\right| - \operatorname{tr}\left\{\boldsymbol{\Psi}^{-1}\tilde{\mathbf{S}}\right\} + 2\operatorname{tr}\left\{\boldsymbol{\Psi}^{-1}\boldsymbol{\Lambda}\hat{\boldsymbol{\beta}}\tilde{\mathbf{S}}\right\} - \operatorname{tr}\left\{\boldsymbol{\Lambda}^\top\boldsymbol{\Psi}^{-1}\boldsymbol{\Lambda}\tilde{\boldsymbol{\Theta}}\right\}\right],$$

where $\tilde{\mathbf{S}} = \sum_{g=1}^{G} \hat{\pi}_g \mathbf{S}_g$, $\tilde{\boldsymbol{\Theta}} = \mathbf{I}_q - \hat{\boldsymbol{\beta}}\hat{\boldsymbol{\Lambda}} + \hat{\boldsymbol{\beta}}\tilde{\mathbf{S}}\hat{\boldsymbol{\beta}}^\top$, and $\hat{\boldsymbol{\beta}} = \hat{\boldsymbol{\Lambda}}^\top\left(\hat{\boldsymbol{\Lambda}}\hat{\boldsymbol{\Lambda}}^\top + \hat{\boldsymbol{\Psi}}\right)^{-1}$.

Differentiating $\mathcal{Q}\left(\boldsymbol{\Lambda}, \boldsymbol{\Psi}\right)$ w.r.t. $\boldsymbol{\Lambda}$ and $\boldsymbol{\Psi}^{-1}$, respectively, gives the score functions

$$\mathcal{S}_1\left(\boldsymbol{\Lambda}, \boldsymbol{\Psi}\right) = \frac{\partial}{\partial \boldsymbol{\Lambda}}\mathcal{Q}_2\left(\boldsymbol{\Lambda}, \boldsymbol{\Psi}\right) = N\left[\boldsymbol{\Psi}^{-1}\tilde{\mathbf{S}}\hat{\boldsymbol{\beta}}^\top - \boldsymbol{\Psi}^{-1}\boldsymbol{\Lambda}\tilde{\boldsymbol{\Theta}}\right], \tag{4.3.6}$$

$$\mathcal{S}_2\left(\boldsymbol{\Lambda}, \boldsymbol{\Psi}\right) = \frac{\partial}{\partial \boldsymbol{\Psi}^{-1}}\mathcal{Q}_2\left(\boldsymbol{\Lambda}, \boldsymbol{\Psi}\right) = \frac{N}{2}\left[\boldsymbol{\Psi} - \tilde{\mathbf{S}}^\top + 2\boldsymbol{\Lambda}\hat{\boldsymbol{\beta}}\tilde{\mathbf{S}} - \boldsymbol{\Lambda}\tilde{\boldsymbol{\Theta}}^\top\boldsymbol{\Lambda}^\top\right]. \tag{4.3.7}$$

Solving $\mathcal{S}_1\left(\hat{\boldsymbol{\Lambda}}^{\text{new}}, \hat{\boldsymbol{\Psi}}^{\text{new}}\right) = 0$ and $\operatorname{diag}\left\{\mathcal{S}_2\left(\hat{\boldsymbol{\Lambda}}^{\text{new}}, \hat{\boldsymbol{\Psi}}^{\text{new}}\right)\right\} = 0$ yields

$$\hat{\boldsymbol{\Lambda}}^{\text{new}} = \tilde{\mathbf{S}}\hat{\boldsymbol{\beta}}^\top\tilde{\boldsymbol{\Theta}}^{-1},$$

and

$$\begin{aligned}\hat{\boldsymbol{\Psi}}^{\text{new}} =& \operatorname{diag}\left\{\tilde{\mathbf{S}} - 2\hat{\boldsymbol{\Lambda}}^{\text{new}}\hat{\boldsymbol{\beta}}\tilde{\mathbf{S}} + \hat{\boldsymbol{\Lambda}}^{\text{new}}\tilde{\boldsymbol{\Theta}}^\top\left(\tilde{\mathbf{S}}\hat{\boldsymbol{\beta}}^\top\tilde{\boldsymbol{\Theta}}^{-1}\right)^\top\right\} \\ =& \operatorname{diag}\left\{\tilde{\mathbf{S}} - \hat{\boldsymbol{\Lambda}}^{\text{new}}\hat{\boldsymbol{\beta}}\tilde{\mathbf{S}}\right\}.\end{aligned}$$

Therefore, the algorithm for the AECM for model CCU is presented in Algorithm 2 (McNicholas, 2016).

Algorithm 2 AECM Algorithm for PGMM Model CCU

```
initialize ẑ_ig
initialize π̂_g, μ̂_g, S̃, Λ̂, Ψ̂
while convergence criterion not met do
    update π̂_g, μ̂_g
    if not iteration 1 then
        update ẑ_ig
    end if
    compute S̃, β̂, Θ̃
    update Λ̂^new, Ψ̂^new
    update ẑ_ig
    check convergence criterion
    Λ̂ ← Λ̂^new, Ψ̂ ← Ψ̂^new
end while
```

Initializing the AECM

As described earlier in Section 4.2.1, there are three methods for initial parameter estimates: random, k-means (default), and user-specified. The component membership estimates $\hat{z}_{ig}$ are found easily enough using one of these methods, allowing the calculation $\hat{\pi}_g$ and $\hat{\boldsymbol{\mu}}_g$ as given in (4.3.11) and (4.3.12), respectively.

The matter of initializing stage 2 of the AECM with estimates of $\hat{\boldsymbol{\Lambda}}$ and $\hat{\boldsymbol{\Psi}}$ requires an approach outlined in McNicholas and Murphy (2008). The suggested approach is the eigen-decomposition of $\tilde{\mathbf{S}}$. First, the $\hat{z}_{ig}$ are used to intitalize $\tilde{\mathbf{S}}$ using $N_g = \sum_{g=1}^{G} \hat{z}_{ig}$, $\tilde{\mathbf{S}} = \sum_{g=1}^{G} \hat{\pi}_g \mathbf{S}_g$, (4.3.3), (4.3.11), and (4.3.12).

As shown in McNicholas (2016), after $\tilde{\mathbf{S}}$ is intialized, using eigen-decomposition results in $\tilde{\mathbf{S}} = \mathbf{PDP}^{-1}$. Initial values for $\boldsymbol{\Lambda}$ and $\boldsymbol{\Psi}$ are then calculated as $\hat{\boldsymbol{\Lambda}} = \mathbf{dP}$ and

$\hat{\mathbf{\Psi}} = \text{diag}\left\{\tilde{\mathbf{S}} - (\mathbf{dP})\,(\mathbf{dP})^\top\right\}$, where $\mathbf{d}$ is element wise square root of the diagonal of $\mathbf{D}$.

For other members of the pgmm family, similar approaches are taken. Some of these are illustrated in McNicholas and Murphy (2008) and McNicholas *et al.* (2010).

4.3.2 Expanded Parsimonious Gaussian Mixture Models

The original family of 8 PGMMs was expanded in McNicholas and Murphy (2010) by parametrizing $\mathbf{\Psi}_g$. In this expanded family, $\mathbf{\Psi}_g = \omega_g \mathbf{\Delta}_g$, where $\omega_g \in \mathbb{R}^+$ and $\mathbf{\Delta}_g$ is a diagonal matrix with $|\mathbf{\Delta}_g| = 1$. This changes the covariance structure for each component to $\mathbf{\Sigma}_g = \mathbf{\Lambda}_g \mathbf{\Lambda}_g^\top + \omega_g \mathbf{\Delta}_g$.

This creates four possible constraints: (1) $\mathbf{\Lambda}_g = \mathbf{\Lambda}$, (2) $\mathbf{\Psi}_g = \mathbf{\Psi}$, (3) $\mathbf{\Psi}_g = \boldsymbol{\psi}_g \mathbf{I}_p$, and (4) $\mathbf{\Psi}_g = \omega_g \mathbf{\Delta}_g$. All legitimate combinations of these four constraints create a model, of which there are 12. This family of 12 models is the expanded PGMM family, shown in Table 4.2.

Parameter Estimation in the Expanded PGMM Family

This topic is covered in McNicholas (2016) and McNicholas and Murphy (2010). For the 8 models in the expanded PGMM (EPGMM) family which have PGMM analogues (see Section 4.3.1), parameter estimates can be obtained by using the methods described in the Section 4.3.1 and then computing the following updates for ω_g and $\mathbf{\Delta}_g$.

$$\hat{\omega}_g = \left|\hat{\mathbf{\Psi}}_g\right|^{1/p}, \tag{4.3.8}$$

Table 4.2: The covariance structures and nomenclature for each member of the expanded PGMM family, along with the name of the equivalent member of the PGMM family (Table 4.1), where applicable.

Expanded PGMM Nomenclature						
$\boldsymbol{\Lambda}_g = \boldsymbol{\Lambda}$	$\boldsymbol{\Delta}_g = \boldsymbol{\Delta}$	$\omega_g = \omega$	$\boldsymbol{\Delta}_g = \mathbf{I}_p$	PGMM Notation	$\boldsymbol{\Sigma}_g$	Number of Covariance Parameters
C	C	C	C	CCC	$\boldsymbol{\Lambda}\boldsymbol{\Lambda}^\top + \omega\mathbf{I}_p$	$[pq - q(q-1)/2] + 1$
C	C	U	C	CUC	$\boldsymbol{\Lambda}\boldsymbol{\Lambda}^\top + \omega_g\mathbf{I}_p$	$[pq - q(q-1)/2] + G$
U	C	C	C	UCC	$\boldsymbol{\Lambda}_g\boldsymbol{\Lambda}_g^\top + \omega\mathbf{I}_p$	$G[pq - q(q-1)/2] + 1$
U	C	U	C	UUC	$\boldsymbol{\Lambda}_g\boldsymbol{\Lambda}_g^\top + \omega_g\mathbf{I}_p$	$G[pq - q(q-1)/2] + G$
C	C	C	U	CCU	$\boldsymbol{\Lambda}\boldsymbol{\Lambda}^\top + \omega\boldsymbol{\Delta}$	$[pq - q(q-1)/2] + p$
C	C	U	U	-	$\boldsymbol{\Lambda}\boldsymbol{\Lambda}^\top + \omega_g\boldsymbol{\Delta}$	$[pq - q(q-1)/2] + [G + (p-1)]$
U	C	C	U	UCU	$\boldsymbol{\Lambda}_g\boldsymbol{\Lambda}_g^\top + \omega\boldsymbol{\Delta}$	$G[pq - q(q-1)/2] + p$
U	C	U	U	-	$\boldsymbol{\Lambda}_g\boldsymbol{\Lambda}_g^\top + \omega_g\boldsymbol{\Delta}$	$G[pq - q(q-1)/2] + [G + (p-1)]$
C	U	C	U	-	$\boldsymbol{\Lambda}\boldsymbol{\Lambda}^\top + \omega\boldsymbol{\Delta}_g$	$[pq - q(q-1)/2] + [1 + G(p-1)]$
C	U	U	U	CUU	$\boldsymbol{\Lambda}\boldsymbol{\Lambda}^\top + \omega_g\boldsymbol{\Delta}_g$	$[pq - q(q-1)/2] + Gp$
U	U	C	U	-	$\boldsymbol{\Lambda}_g\boldsymbol{\Lambda}_g^\top + \omega\boldsymbol{\Delta}_g$	$G[pq - q(q-1)/2] + [1 + G(p-1)]$
U	U	U	U	UUU	$\boldsymbol{\Lambda}_g\boldsymbol{\Lambda}_g^\top + \omega_g\boldsymbol{\Delta}_g$	$G[pq - q(q-1)/2] + Gp$

and

$$\hat{\boldsymbol{\Delta}}_g = \frac{\hat{\boldsymbol{\Psi}}_g}{\left|\hat{\boldsymbol{\Psi}}_g\right|^{1/p}}. \tag{4.3.9}$$

Note that $\hat{\omega}_g\hat{\boldsymbol{\Delta}}_g = \hat{\boldsymbol{\Psi}}_g$. For the remaining four models of the EPGMM family parameter estimation isn't as straightforward. Since the parametrization $\boldsymbol{\Psi}_g = \omega_g\boldsymbol{\Delta}_g$ requires $|\boldsymbol{\Delta}_g| = 1$, this complicates the second stage of the AECM described earlier. This complication is resolved by using the method of Lagrange multipliers.

The four models which are solely in the EPGMM are CCUU, UCUU, CUCU, and UUCU. As an example, consider model CCUU.

AECM: Stage 1

This derivation will proceed in a manner similar to Section 4.3.1 following the methods and notations in McNicholas and Murphy (2010). In this stage, the complete data are taken to be the observations $\mathbf{x}_1, \ldots, \mathbf{x}_N$ and the unobserved $\mathbf{z}_1, \ldots, \mathbf{z}_N$. In the E-step of stage 1, the z_{ig} are replaced by their conditional expected values:

$$\mathbb{E}\left[Z_{ig} \mid \hat{\boldsymbol{\vartheta}}\right] = \hat{z}_{ig} = \frac{\hat{\pi}_g\phi\left(\mathbf{x}_i \mid \hat{\boldsymbol{\vartheta}}\right)}{\sum_{h=1}^{G}\hat{\pi}_h\phi\left(\mathbf{x}_i \mid \hat{\boldsymbol{\vartheta}}\right)}, \tag{4.3.10}$$

where $\hat{\boldsymbol{\vartheta}} = \left(\pi_g, \hat{\boldsymbol{\mu}}_g, \hat{\boldsymbol{\Lambda}}_g, \hat{\boldsymbol{\Delta}}_g, \hat{\omega}_g\right)$. Using $\hat{z}_{ig}$, $\mathcal{Q}_1$ is computed and then maximized in the CM-step to yield

$$\hat{\pi}_g = \frac{N_g}{N}, \tag{4.3.11}$$

$$\hat{\boldsymbol{\mu}}_g = \frac{\sum_{i=1}^{N}\hat{z}_{ig}\mathbf{x}_i}{N_g}, \tag{4.3.12}$$

where $N_g = \sum_{g=1}^{G}\hat{z}_{ig}$.

AECM: Stage 2

For the stage 2 of the AECM, the complete-data will be taken as the observations $\mathbf{x}_1, \ldots, \mathbf{x}_N$, the unobserved $\mathbf{z}_1, \ldots, \mathbf{z}_N$, and the latent factors $\mathbf{u}_{ig}$, for $i = 1, \ldots, N$ and $g = 1, \ldots, G$. The unobserved $\mathbf{z}_1, \ldots, \mathbf{z}_N$ addressed in stage 1, and the $\mathbf{u}_{ig}$ will be addressed here. Both $\mathbf{u}_{ig}$ and $\mathbf{u}_{ig}\mathbf{u}_{ig}^{\top}$ will be replaced by their expected values:

$$\mathbb{E}\left[\mathbf{U}_{ig} \mid \mathbf{x}_i, \hat{\boldsymbol{\mu}}_g, \hat{\boldsymbol{\Lambda}}_g, \hat{\omega}_g, \hat{\boldsymbol{\Delta}}_g\right] = \hat{\boldsymbol{\beta}}_g\left(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_g\right),$$

$$\mathbb{E}\left[\mathbf{U}_{ig}\mathbf{U}_{ig}^{\top}\right] = \mathbf{I}_q - \hat{\boldsymbol{\beta}}_g\hat{\boldsymbol{\Lambda}}_g + \hat{\boldsymbol{\beta}}_g\left(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_g\right)\left(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_g\right)^{\top}\hat{\boldsymbol{\beta}}_g^{\top},$$

where $\hat{\boldsymbol{\beta}}_g = \hat{\boldsymbol{\Lambda}}_g^{\top}\left(\hat{\boldsymbol{\Lambda}}_g\hat{\boldsymbol{\Lambda}}_g^{\top} + \hat{\omega}_g\hat{\boldsymbol{\Delta}}_g\right)^{-1}$. These expected values then yield the expected value of the complete-data log-likelihood ($\mathcal{Q}_2$) which can now be maximized in the CM-step to yield the remaining parameter estimates. This will depend specifically on the model. For the CCUU model, the constraints are $\boldsymbol{\Lambda}_g = \boldsymbol{\Lambda}$ and $\boldsymbol{\Delta}_g = \boldsymbol{\Delta}$. In this case,

$$\begin{aligned}\mathcal{Q}_2\left(\boldsymbol{\Lambda}, \omega_g, \boldsymbol{\Delta}\right) = C + \frac{1}{2}\sum_{g=1}^{G} N_g\Big[p\log\omega_g^{-1} + \log\left|\boldsymbol{\Delta}^{-1}\right| - \omega_g^{-1}\operatorname{tr}\left\{\boldsymbol{\Delta}^{-1}\mathbf{S}_g\right\} \\ + 2\omega_g^{-1}\operatorname{tr}\left\{\boldsymbol{\Delta}^{-1}\boldsymbol{\Lambda}\hat{\boldsymbol{\beta}}_g\mathbf{S}_g\right\} - \omega_g^{-1}\operatorname{tr}\left\{\boldsymbol{\Lambda}^{\top}\boldsymbol{\Delta}^{-1}\boldsymbol{\Lambda}\boldsymbol{\Theta}_g\right\}\Big],\end{aligned}$$

where C is constant with respect to $\boldsymbol{\Lambda}$, ω_g, and $\boldsymbol{\Delta}$ and $\boldsymbol{\Theta}_g = \mathbf{I}_q - \hat{\boldsymbol{\beta}}_g\hat{\boldsymbol{\Lambda}} + \hat{\boldsymbol{\beta}}_g\mathbf{S}_g\hat{\boldsymbol{\beta}}_g^{\top}$.

In order to maximize $\mathcal{Q}_2$ with respect to $\boldsymbol{\Lambda}$, ω_g, and $\boldsymbol{\Delta}$, the method of Lagrange

multipliers is employed. The Lagrange function is

$$\mathbf{L}\left(\boldsymbol{\Lambda}, \omega_g, \boldsymbol{\Delta}, \kappa\right) = \mathcal{Q}(\boldsymbol{\Lambda}, \omega_g, \boldsymbol{\Delta}) - \kappa(|\boldsymbol{\Delta}| - 1).$$

Similar to McNicholas (2016), taking the partial derivatives yields

$$\begin{aligned}
\mathcal{S}_1\left(\boldsymbol{\Lambda}, \omega_g, \boldsymbol{\Delta}, \kappa\right) =& \frac{\partial \mathbf{L}}{\partial \boldsymbol{\Lambda}} = \sum_{g=1}^{G} \frac{N_g}{\omega_g}\left[\boldsymbol{\Delta}^{-1}\mathbf{S}_g\hat{\boldsymbol{\beta}}_g^{\top} - \boldsymbol{\Delta}^{-1}\boldsymbol{\Lambda}\boldsymbol{\Theta}_g\right], \\
\mathcal{S}_2\left(\boldsymbol{\Lambda}, \omega_g, \boldsymbol{\Delta}, \kappa\right) =& \frac{\partial \mathbf{L}}{\partial \omega_g^{-1}} \\
=& \frac{N_g}{2}\left[p\omega_g - \operatorname{tr}\left\{\boldsymbol{\Delta}^{-1}\mathbf{S}_g\right\} + 2\operatorname{tr}\left\{\boldsymbol{\Delta}^{-1}\boldsymbol{\Lambda}\hat{\boldsymbol{\beta}}_g\mathbf{S}_g\right\} - \operatorname{tr}\left\{\boldsymbol{\Delta}^{-1}\boldsymbol{\Lambda}\boldsymbol{\Theta}_g\boldsymbol{\Lambda}^{\top}\right\}\right], \\
\mathcal{S}_3\left(\boldsymbol{\Lambda}, \omega_g, \boldsymbol{\Delta}, \kappa\right) =& \frac{\partial \mathbf{L}}{\partial \boldsymbol{\Delta}^{-1}} \\
=& \frac{1}{2}\sum_{g=1}^{G} N_g\left[\boldsymbol{\Delta} - \omega_g^{-1}\mathbf{S}_g + 2\omega_g^{-1}\left(\boldsymbol{\Lambda}\hat{\boldsymbol{\beta}}_g\mathbf{S}_g\right)^{\top} - \omega_g^{-1}\boldsymbol{\Lambda}\boldsymbol{\Theta}_g\boldsymbol{\Lambda}^{\top}\right] + \kappa\left|\boldsymbol{\Delta}\right|\boldsymbol{\Delta}, \\
\mathcal{S}_4\left(\boldsymbol{\Lambda}, \omega_g, \boldsymbol{\Delta}, \kappa\right) =& \frac{\partial \mathbf{L}}{\partial \kappa} \\
=& |\boldsymbol{\Delta}| - 1.
\end{aligned}$$

Setting the score functions to zero: $\mathcal{S}_1\left(\hat{\boldsymbol{\Lambda}}^{\text{new}}, \hat{\omega}_g, \hat{\boldsymbol{\Delta}}, \kappa\right) = 0$, $\mathcal{S}_2\left(\hat{\boldsymbol{\Lambda}}^{\text{new}}, \hat{\omega}_g^{\text{new}}, \hat{\boldsymbol{\Delta}}, \kappa\right) = 0$, and $\mathcal{S}_3\left(\hat{\boldsymbol{\Lambda}}^{\text{new}}, \hat{\omega}_g^{\text{new}}, \hat{\boldsymbol{\Delta}}^{\text{new}}, \kappa\right) = 0$ yields the estimates

$$\hat{\boldsymbol{\Lambda}}^{\text{new}} = \left[\sum_{g=1}^{G}\frac{N_g}{\hat{\omega}_g}\mathbf{S}_g\hat{\boldsymbol{\beta}}_g^{\top}\right]\left[\sum_{g=1}^{G}\frac{N_g}{\hat{\omega}_g}\boldsymbol{\Theta}_g\right]^{-1},$$

$$\hat{\omega}_g^{\text{new}} = \frac{1}{p}\operatorname{tr}\left\{\hat{\boldsymbol{\Delta}}^{-1}\left[\mathbf{S}_g - 2\hat{\boldsymbol{\Lambda}}^{\text{new}}\hat{\boldsymbol{\beta}}_g\mathbf{S}_g + \hat{\boldsymbol{\Lambda}}^{\text{new}}\boldsymbol{\Theta}_g\left(\hat{\boldsymbol{\Lambda}}^{\text{new}}\right)^{\top}\right]\right\},$$

$$\hat{\boldsymbol{\Delta}}^{\text{new}} = \frac{1}{N+2\kappa}\text{diag}\left\{\sum_{g=1}^{G}\frac{N_g}{\hat{\omega}_g^{\text{new}}}\left[\mathbf{S}_g - 2\hat{\boldsymbol{\Lambda}}^{\text{new}}\hat{\boldsymbol{\beta}}_g\mathbf{S}_g + \hat{\boldsymbol{\Lambda}}^{\text{new}}\boldsymbol{\Theta}_g\left(\hat{\boldsymbol{\Lambda}}^{\text{new}}\right)^{\top}\right]\right\}.$$

However, it is required that $\left|\hat{\boldsymbol{\Delta}}^{\text{new}}\right| = 1$. To ensure this property it is required that

$$N + 2\kappa = \left(\prod_{i=1}^{p}\xi_i\right)^{\frac{1}{p}},$$

where ξ_i are the diagonal elements of the matrix

$$\sum_{g=1}^{G}\frac{N_g}{\hat{\omega}_g^{\text{new}}}\left[\mathbf{S}_g - 2\hat{\boldsymbol{\Lambda}}^{\text{new}}\hat{\boldsymbol{\beta}}_g\mathbf{S}_g + \hat{\boldsymbol{\Lambda}}^{\text{new}}\boldsymbol{\Theta}_g\left(\hat{\boldsymbol{\Lambda}}^{\text{new}}\right)^{\top}\right].$$

Solving for κ gives $\kappa = \frac{1}{2}\left[\left(\prod_{i=1}^{p}\xi_i\right)^{\frac{1}{p}} - N\right]$. The estimates for the other three EPGMM models are derived in a similar manner (McNicholas and Murphy, 2010).

4.3.3 MDA Framework

The primary aim of MDA is to ascertain the component membership vectors, denoted $\mathbf{z}_1, \ldots, \mathbf{z}_N$, where $\mathbf{z}_i = (z_{i1}, \ldots, z_{iG})$ and $z_{ig} = 1$ if $\mathbf{x}_i$ belongs to group g and $z_{ig} = 0$ otherwise. The observations are denoted $\mathbf{T} = \{(y_i, \mathbf{x}_i)|i = 1, \ldots, N\}$, where only n of the memberships are known a priori. These can be arranged $(y_1, \mathbf{x}_1), \ldots, (y_n, \mathbf{x}_n), \mathbf{x}_{n+1}, \ldots, \mathbf{x}_N$ without loss of generality. Then, the first n observations will be used to carry out the estimation of $\mathbf{z}_{n+1}, \ldots, \mathbf{z}_N$.

In MDA, a mixture of factor analyzers is fitted for every observed class. In an attempt to detect sub-classes, a model-based cluster analysis is first carried out for each of these observed classes, though sometimes this is not applicable. As shown

by McNicholas (2016), this procedure is outlined by Algorithm 3.

Algorithm 3 Model-Based Discriminant Analysis

for g in 1 to G **do**
  carry out a model-based clustering for $\mathbf{x}_1, \ldots, \mathbf{x}_n$ in class g
  choose a G_g-component model
  record corresponding labels LAB_g
end for
compute $G_{\text{new}} = \sum_{g=1}^{G} G_i$
fit a G_{new}-component mixture to $\mathbf{x}_1, \ldots, \mathbf{x}_n$ using labels $\text{LAB}_1, \ldots, \text{LAB}_G$
record resulting parameter estimates $\hat{\boldsymbol{\vartheta}}$
for $i = n + 1$ to N **do**
  for g in 1 to G_{new} **do**
    compute $\hat{z}_{ig}$
  end for
  assign $\mathbf{x}_i$ to the class corresponding to component $\text{argmax}_g\, \hat{z}_{ig}$
end for

At the end of the model-fitting procedure described in Section 4.3.2, the resultant estimates take the form of conditional expectations (see McNicholas, 2016)

$$\mathbb{E}\left[Z_{ig} \mid \hat{\boldsymbol{\vartheta}}\right] = \hat{z}_{ig} = \frac{\hat{\pi}_g \phi\left(\mathbf{x}_i \mid \hat{\boldsymbol{\vartheta}}\right)}{\sum_{h=1}^{G} \hat{\pi}_h \phi\left(\mathbf{x}_i \mid \hat{\boldsymbol{\vartheta}}\right)}. \tag{4.3.13}$$

The $\hat{z}_{ig} \in [0, 1]$ and can be interpreted as probabilities of the observation $\mathbf{x}_i$ belonging to component g. As such, they are called soft classifications, which can be hardened into maximum a posteriori classifications (i.e., $\text{MAP}\{\hat{z}_{ig}\}$), such that

$$\text{MAP}\{\hat{z}_{ig}\} = \begin{cases} 1, & \text{if } g = \text{argmax}_h\{\hat{z}_{ih}\}, \\ 0, & \text{otherwise.} \end{cases}$$

4.4 MixForests

4.4.1 Partitioning the Data

The observations $\mathbf{x}_1, \ldots, \mathbf{x}_N$ were partitioned into a training set and a holdout set, using a holdout percentage of 20%. The partitioning was done carefully, to get an accurate representation of each of the classes represented in the data. This was done by randomly selecting 20% of the observations from each class, to get a stratified sample. In this way, the test error calculation should better reflect reality.

4.4.2 Generating a MixForest

A bootstrap replicate is first generated from the training data by making N_{train} random draws on the training set, where N_{train} is the number of observations in the training set. Then a random selection of r features from the p features in the bootstrap replicate are used to fit a mixture of factor analyzers. K such mixtures are generated to create the mixForest. This will leave roughly 37% of the observations out-of-bag for each bootstrap replicate. These out-of-bag observations will be used to find the correct number of randomly selected features for the final inferences.

In addition to this, for every classifier (mixture of factor analyzers) in the mixForest, a range of values for q is attempted for each of the model types listed in Table 4.2. This range of q is chosen so that

$$(p-q)^2 > (p+q),$$

in order to ensure that there is a reduction in the number of free covariance parameters (Lawley and Maxwell, 1962).

Furthermore, a range of values for G_{new} is attempted (refer to Algorithm 3). For a particular bootstrap replicate, all combinations of model type, values of q, and G_{new} are fitted, and the mixture of factor analyzers with the maximum BIC is used in the mixForest.

4.4.3 Random Feature Selection

As described in Section 2.3.4, the number of randomly selected features (r) is the most important parameter in the generation of the mixForest. It will be shown later that changing this one parameter will greatly affect the test error produced by the mixForests fitted in Section 4.5.

Unfortunately, it is not reasonable to assume that there is a performance pattern based solely on the magnitude of r, so it will be necessary to fit p mixForests for each dataset before the value of r can be tuned.

4.4.4 Exploratory Fittings vs. Full Fitting

To save on computation time, an exploratory fitting using specific of values for r is generated first. For example, when using a dataset with $p = 30$, three mixForests with $r = 10, 20, 30$ and a relatively large value of K are fitted allowing insight into the convergence properties of the OOB error sequences for different values of r.

This will allow an appropriate choice of K for a complete set of mixForests to be generated for $r = 1, \ldots, 30$. The second full fitting is necessary since it is not

apparent which values of r will produce a mixForest with relatively low OOB errors.

4.4.5 Aggregation

For a single mixture of factor analyzers, an observation $\mathbf{x}_i$ is classified using (4.3.13). The class g, for which MAP$[\hat{z}_{ig}] = 1$, is considered to be this classifier's vote for observation $\mathbf{x}_i$.

The votes are tallied for all the classifiers in the mixForest using a majority vote, and the class g with the most votes is considered the predicted class of the mixForest for observation $\mathbf{x}_i$. This process is the same regardless of whether $\mathbf{x}_i$ is an OOB observation, or if it belongs to the testing data, or if it belongs to the training data.

4.5 Data Analyses and Illustrations

4.5.1 Forest Type Mapping Data

The original data was gathered and used by Johnson *et al.* (2012). Part of the data were collected during a remote sensing study of forested areas in Ibaraki Prefecture, Japan spanning an area of approximately 13 km x 12 km. Orthorectified ASTER imagery was acquired using the ASTER (Advanced Spaceborne Thermal Emission and Reflection Radiometer) sensors on board the Terra satellite launched by NASA in 1999 (Wikipedia, 2019).

The goal of the study was to classify images into one of four land cover types: (1) *Cryptomeria japonica* (Sugi, or Japanese Cedar), (2) *Chamaecyparis obtusa* (Hinoki, or Japanese Cypress), (3) mixed deciduous broadleaf natural forest, and (4) other

land use/land cover types. So, for this data there are four classes ($G = 4$). Each forest type has a different economic usage as well as environmental profile, so classification of these areas is important for the economy and for ecological study.

This dataset consists of 523 observations with each feature vector consisting of 9 spectral bands (ASTER) and 18 similarity measures (based on IDW (inverse distance weighted interpolated values), for a total of 27 features. The original classification was performed using support vector machines (SVM) and attained an overall accuracy of 85.9%.

To begin the analysis of the data, the mixForest requires that two parameters be addressed. The first is the ensemble size. The second is the number of randomly selected features for which the models in the mixForest are fitted. Thus, a preliminary fitting of the data is performed in order to gauge the properties of convergence. In this endeavour, the main inferential quantity is OOB error, which estimates test error.

For the exploratory fitting, mixForests with numbers of randomly selected features in $\{5, 10, 15, 20, 25\}$ were generated with an ensemble size of 200. The OOB error plot are given in Figure 4.1.

From Figure 4.1, it can be seen that the OOB error of each mixForest stabilizes quickly at approximately $K = 17$ learners. After this point, there is no deviation in the OOB error of the mixForest. The tail end of Figure 4.1 is shown in Figure 4.2 for further examination. Clearly, there are notable and significant differences in the OOB error produced by mixForests based on different numbers of randomly selected features.

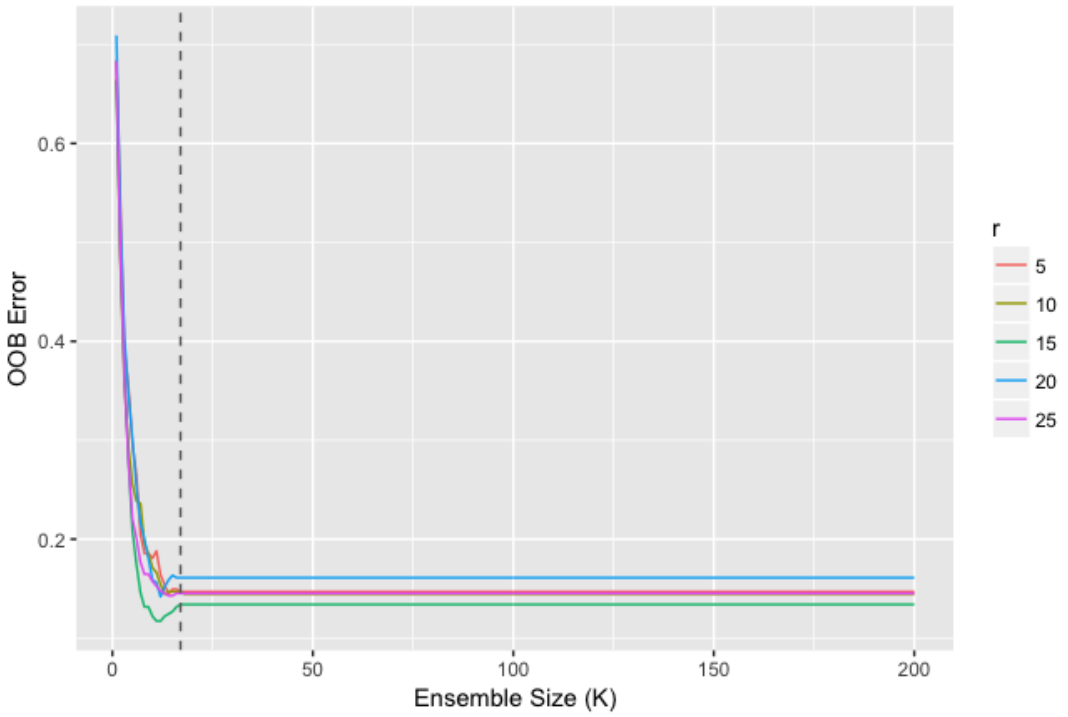

Figure 4.1: OOB error graph of the forest type mapping data, with $r \in \{5, 10, 15, 20, 25\}$.

For the wide range of the number of randomly selected features r in the exploratory fitting, it can be seen that convergence of OOB error is rapid and stable for all choices of r. This means that when a more thorough fitting is performed, K can be much smaller than before, saving computation time.

Next, we examine the sequential differences in the OOB error sequences and record the first value of K for which the difference is less than 10^{-5} (see Table 4.11). Since the OOB error does not deviate after the tolerance is reached, ensemble sizes can be kept small.

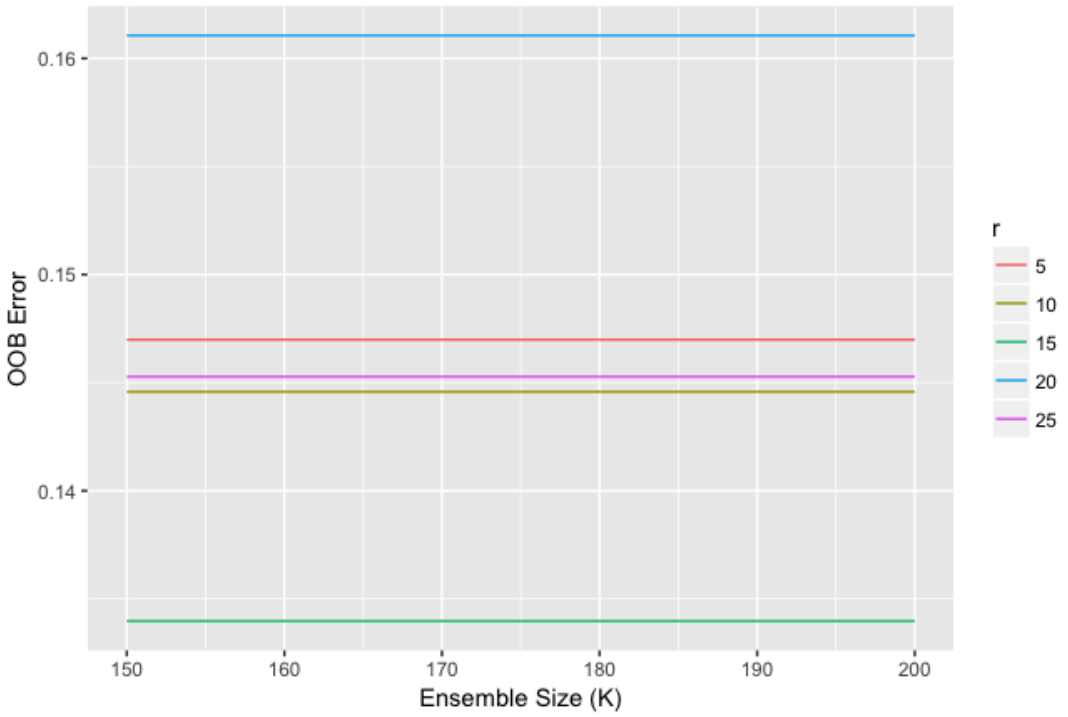

Figure 4.2: OOB error graph of the forest type mapping data, with $r \in \{5, 10, 15, 20, 25\}$ for $K = 150, \ldots, 200$.

Table 4.3: Limiting ensemble size for choices of number of random features.

# of Random Features	Ensemble Size
5	9
10	15
15	9
20	17
25	9

For the second consideration of selecting an optimal value of the number of randomly selected features (r), a second set of 27 mixForests is generated for each of $r = 1, \ldots, 27$ with $K = 75$. As a rule of thumb, except for computation time, there

is no detriment in fitting more models. Figure 4.3 and Figure 4.4 illustrate the corresponding OOB error sequences.

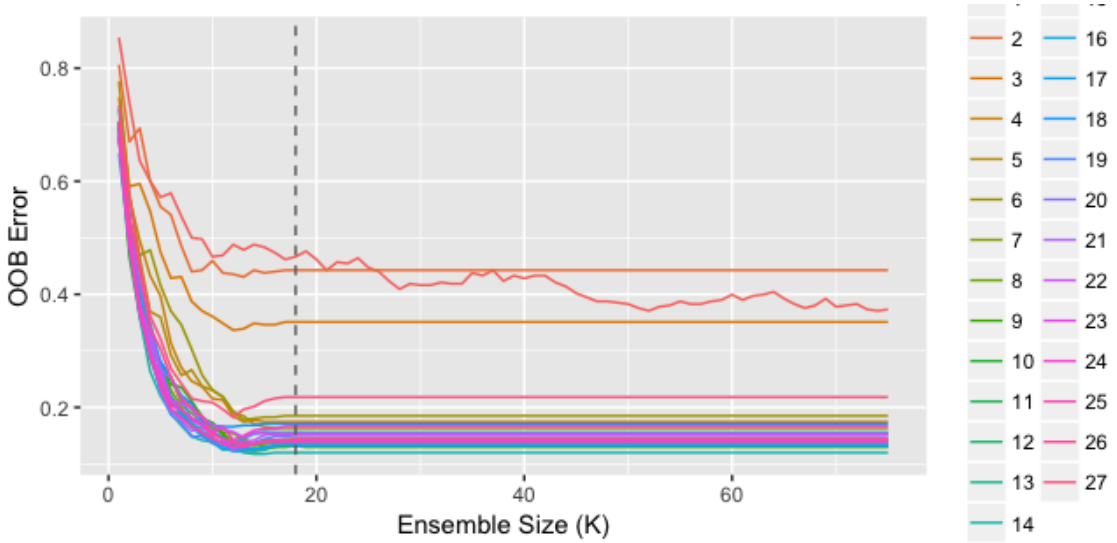

Figure 4.3: OOB error graph of the forest type mapping data, for $r = \{1, \ldots, 27\}$.

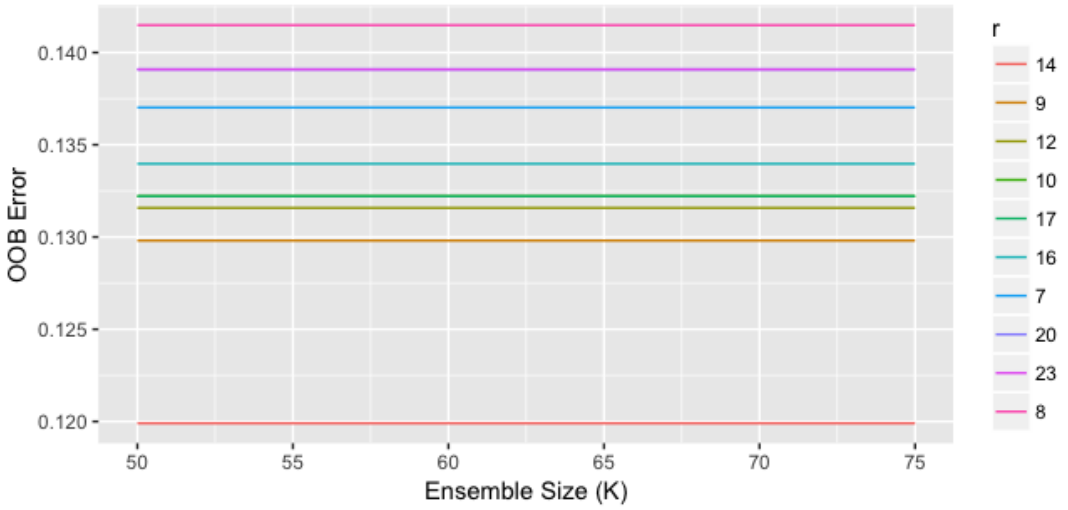

Figure 4.4: OOB error graph of the forest type mapping data, with $r = 7, 8, 9, 10, 12, 14, 16, 17, 20, 23$ and $K = 50, \ldots, 75$.

There are no evident patterns in either graph. For further examination, Table 4.4 and Table 4.5 give the exact values of OOB error for each value of r. The only obvious trend is that any choice of r between 7 and 26 (inclusively) seems to work, though $r = 14$ produces the lowest OOB error.

The final test is a comparison of the test error produced by the mixForest versus MDA. For this comparison, the data were partitioned into a training set (comprised of 80% of the data) and a holdout set (comprised of 20% of the data), where the data were chosen at random and an equal number of observations were taken from each class. The test errors are given in Table 4.6.

All test errors for mixForests with $r = 1, \ldots, 27$ are given in Table 4.7. For $7 \leq r \leq 27$, all test errors appear acceptable, with $r = 14$ producing the best test error. This confirms the previous hypothesis that $r = 14$ would be the best choice, according to the OOB errors shown in Table 4.5.

The confusion matrix for the mixForest with $r = 14$ is shown in Table 4.8.

It is evident that for the forest types mapping data the mixForest technique compares very favourably with MDA, yielding an increase in accuracy of 19.05%. In addition, the mixForest is able to produce variable importance scores (shown in Table 4.9).

From the Table 4.9, it can be seen there are many variables which are significant at the $p = 0.05$ level. These variables are $\{1, 2, 3, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 20, 21, 25\}$, which comprise roughly 63% of the dataset. These represent a good mix between the ASTER band readings and similarity measures.

Finally, to examine the dimensional reduction taking place, a table of the number

Table 4.4: OOB error table of the forest types mapping data.

# of Random Features (r)	OOB Error
1	0.3732
2	0.4426
3	0.3511
4	0.1746
5	0.1631
6	0.1851
7	0.1370
8	0.1415
9	0.1298
10	0.1322
11	0.1439
12	0.1316
13	0.1559
14	0.1199
15	0.1538
16	0.1340
17	0.1322
18	0.1707
19	0.1511
20	0.1391
21	0.1535
22	0.1518
23	0.1391
24	0.1667
25	0.1425
26	0.1446
27	0.2182

Table 4.5: Sorted OOB error table for the forest types mapping data.

# of Random Features (r)	OOB Error
14	0.1199
9	0.1298
12	0.1316
10	0.1322
17	0.1322
16	0.1340
7	0.1370
20	0.1391
23	0.1391
8	0.1415
25	0.1425
11	0.1439
26	0.1446
19	0.1511
22	0.1518
21	0.1535
15	0.1538
13	0.1559
5	0.1631
24	0.1667
18	0.1707
4	0.1746
6	0.1851
27	0.2182
3	0.3511
1	0.3732
2	0.4426

Table 4.6: Test error comparison, using $r = 14$.

	MDA	MixForest
Test Error	0.2762	0.0857

of latent factors the leading mixForest produced is included (Table 4.10).

4.5.2 Ionosphere Data

The primary motive of the original study (Sigillito *et al.* (1989)) is to classify radar returns as either good (suitable for further analysis), or bad, with the aim of studying the physics of the ionosphere. Automated classification of the radar returns will save valuable manpower. Sigillito *et al.* (1989) states the data were collected by the Space Physics Group of the Johns Hopkins University Applied Physics Laboratory using the radar system located in Goose Bay, Labrador.

The original paper describes that the radar transmits a multipulse pattern into the ionosphere, after which a receiver is turned on intermittently and the complex electromagnetic signal is recorded. The complex ACF (auto-correlation function) is created by measuring 17 discrete returns, each of which has a real and an imaginary part (all of which are continuous). The dataset are comprised of 350 observations, with the 34 features described. The class variable is binary.

Sigillito *et al.* (1989) states that a multilayer feedforward neural network (MLFN) used to perform the classification, achieved 100% accuracy on the training data and up to 98% accuracy using the testing data.

As in the previous data analysis, the first step is to determine the minimum

Table 4.7: Test error table of the forest types mapping data.

# of Random Features (r)	Test Error
1	0.3429
2	0.4571
3	0.3524
4	0.1524
5	0.1524
6	0.0952
7	0.1810
8	0.1333
9	0.1143
10	0.1143
11	0.1048
12	0.0952
13	0.2000
14	0.0857
15	0.1619
16	0.1238
17	0.1333
18	0.1238
19	0.1429
20	0.1333
21	0.1714
22	0.1143
23	0.1143
24	0.0952
25	0.1048
26	0.0952
27	0.2286

Table 4.8: Forest types mapping data confusion matrix for mixForest with $r = 14$ and $K = 75$.

		Actual			
		d	h	o	s
Predicted	d	30	0	0	1
	h	0	13	0	2
	o	1	0	17	0
	s	1	4	0	36

ensemble size required to produce a mixForest with optimal results. For the first exploratory fitting of this dataset, mixForests of size $K = 100$ for $r = 5, 10, 15, 20, 25, 30$ randomly selected features were generated. The OOB error plot for these mixForests are shown in Figure 4.5.

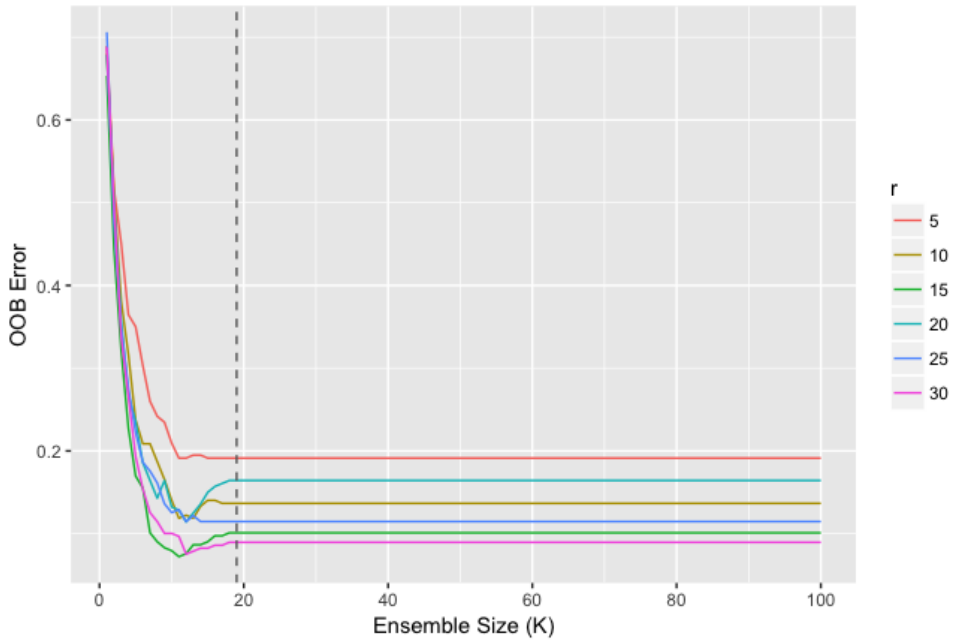

Figure 4.5: OOB error graph of the ionosphere data, with $r = \{5, 10, 15, 20, 25, 30\}$.

Table 4.9: Variable importance table of the forest types mapping data.

Variable	Raw Score	Z Score	P Value
3	46.62	7.75	0.0000
1	30.35	6.49	0.0000
11	46.65	6.42	0.0000
18	39.10	6.38	0.0000
9	42.86	5.29	0.0000
7	52.96	5.24	0.0000
8	22.19	4.91	0.0000
15	22.41	4.31	0.0000
16	52.45	3.86	0.0001
14	30.21	3.59	0.0003
25	15.38	2.95	0.0032
2	47.22	2.63	0.0085
21	7.07	2.43	0.0150
10	26.00	2.39	0.0169
13	13.57	2.23	0.0256
20	5.75	2.19	0.0283
17	24.25	2.15	0.0318
6	27.00	1.43	0.1538
22	3.00	1.41	0.1573
23	1.00	1.41	0.1573
12	24.12	1.40	0.1611
5	35.83	1.38	0.1663
4	13.75	1.36	0.1753
19	4.75	0.79	0.4281
26	2.24	0.74	0.4604
27	6.43	0.72	0.4722
24	3.50	0.69	0.4901

Table 4.10: Number of latent factors (q) used by the mixForest with $r = 14$ in the forest types mapping data.

q	Frequency
6	3
7	6
9	66

In Figure 4.6, the tail end of Figure 4.5 is shown. There are clear and significant differences in OOB error, ranging from 9.0% to 19.1%, though the plot does not indicate a pattern based on values of r.

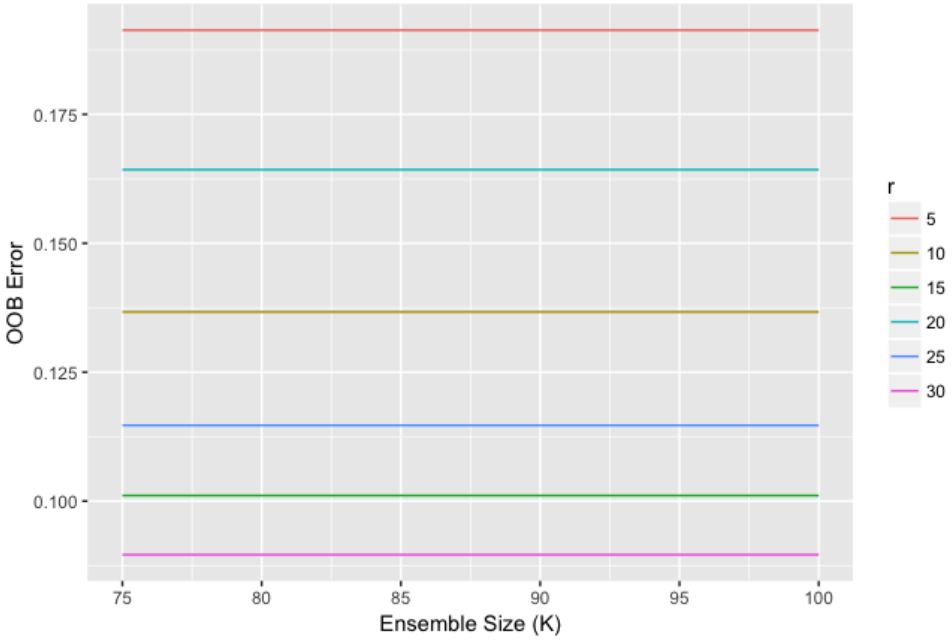

Figure 4.6: Tail of the OOB error graph of the ionosphere data, with $r = \{5, 10, 15, 20, 25, 30\}$.

Looking at Figure 4.5, there is quick stabilization of the OOB error plots for every

value of r. As classifiers are fitted and added to the MixForest using the sequence of bootstrap replicates, the OOB error of the MixForest is updated. The first value of K for which the difference between two sequential OOB error estimates is less than 10^{-5} is shown in Table 4.11. A second set of mixForests were fitted using $r = 1, \ldots, 33$

Table 4.11: Limiting ensemble size for choices of number of random features for the ionosphere data.

# of Random Features	Ensemble Size
5	12
10	7
15	14
20	19
25	15
30	10

and $K = 50$. The OOB error plots are shown in Figure 4.7 and Figure 4.8. Exact values are given in the Appendix in Table A.1, Table A.2, Table A.3, and Table A.4.

From the OOB error plots and tables, there is an optimal value between $r = 8, \ldots, 12$. The OOB error for $r = 10$ may be an anomaly for this particular set of training data. Also, there seems to be a higher range ($r = 25, \ldots, 31$) that performs quite well. It is difficult to deduce exactly which value of r will perform the best in practice. This is not an exact science, and the matter is not as clear cut as in the previous data analysis.

For the purpose of illustration, further analysis will be done using several values of r, though $r = 12$ seems to perform the best. Arguably, the only way to select the

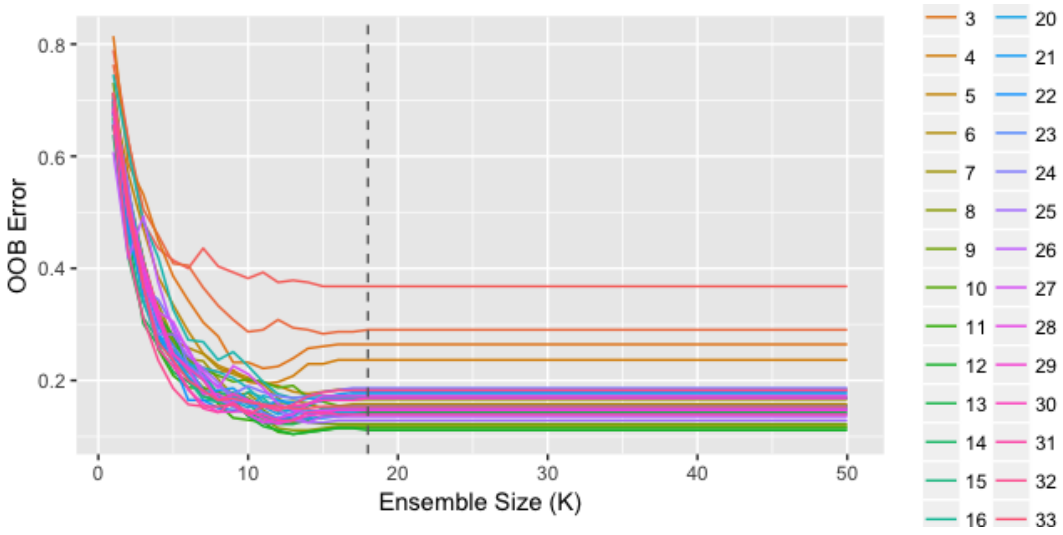

Figure 4.7: OOB error graph of the ionosphere data, with $r = 1, \ldots, 33$.

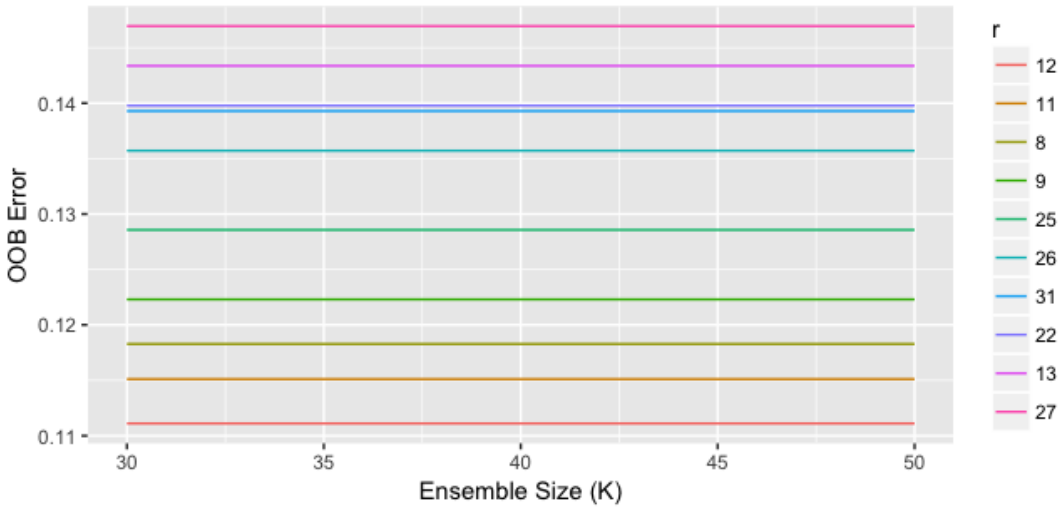

Figure 4.8: OOB error graph of the ionosphere data: 10 lowest sequences.

correct value of r is to fit mixForests for all possible values of r. The test errors of several mixForests and MDA are given in Table 4.12.

Table 4.12 yields interesting results, as the purported favourite (mixForest with

Table 4.12: Test error comparison of models fitted to the ionosphere data.

Model	Test Error
MDA	0.0286
MixForest ($r = 8$)	0.0857
MixForest ($r = 9$)	0.0714
MixForest ($r = 10$)	0.1000
MixForest ($r = 11$)	0.0714
MixForest ($r = 12$)	0.0429
MixForest ($r = 25$)	0.0000
MixForest ($r = 26$)	0.0143

$r = 12$) did not perform as well as expected. Additionally, the spike in OOB error at $r = 10$ does not seem like an anomaly. The final point is that the mixForests for $r = 25$ and $r = 26$ performed the best. Confusion matrices for $r = 12, 25, 26$ are given in Table 4.13, Table 4.14 and Table 4.15.

Table 4.13: Ionosphere data confusion matrix for mixForest with $r = 12$ and $K = 50$.

		Actual	
		bad	good
Predicted	bad	23	1
	good	2	44

Variable importance scores are shown in Table A.5 and Table A.6 in the Appendix. The results are very surprising since there are very few variables that are significant, but the mixForests fitted with a higher value of r performed better.

Finally, Table 4.16 shows the number of latent factors that were utilized by the

Table 4.14: Ionosphere data confusion matrix for mixForest with $r = 25$ and $K = 50$.

		Actual	
		bad	good
Predicted	bad	25	0
	good	0	45

Table 4.15: Ionosphere data confusion matrix for mixForest with $r = 26$ and $K = 50$.

		Actual	
		bad	good
Predicted	bad	25	1
	good	0	44

mixForest with $r = 25$. As can be seen, $q = 2$ is almost exclusively used by every learner in the ensemble, indicating that great dimensional reduction occurred.

Table 4.16: Number of latent factors (q) used by the mixForest with $r = 25$ for the ionosphere data.

q	Frequency
1	5
2	39
3	1
4	1
5	1
12	2
15	1

4.5.3 Pen-based Handwritten Digit Recognition Data

The original source of this data is Alimoglu *et al.* (1996). A digit database was created by collecting 250 samples from 44 authors of the digits $0, 1, \ldots, 9$. This data was collected using a WACOM PL-100V pressure sensitive tablet with an integrated LCD display and a cordless stylus. The tablet records (x, y) coordinate information, which is then resampled using simple linear interpolation between pairs of points. The data is represented as a sequence of 8 points, with 2 coordinates each (x and y), for a total of 16 features in each observation. The dataset consists of 10992 observations, 1 target attribute (with 10 classes), and 16 features.

As was demonstrated in the previous analyses, the ensembles do not have to be too large before the error rates stabilize and converge. An exploratory fitting was first performed with $r = 5, 10, 15$ of size $K = 50$ in an effort to ensure similar convergent behaviour is present. These results are shown in Figure 4.9. A closer look at the tail end of the plot is show in Figure 4.10.

From Figure 4.9 and Figure 4.10, the OOB error stabilizes rapidly for all values of r into clear and significant levels. Next, a more complete set of fittings is done for $r = 1, \ldots, 16$ and $K = 30$. The OOB error sequences for these fittings are shown in Figure 4.11 and Figure 4.12.

From Figure 4.11 and Figure 4.12, it is seen for for a value of $r = 2$, the ensemble does not seem to reach a convergent OOB error, though it appears bounded. All other values of r are performing as expected. Also, there appears to be one ensemble which is performing much better than the rest.

From Table 4.17, there is one sweet spot for $r = 6, 7, 8$ and another for $r =$

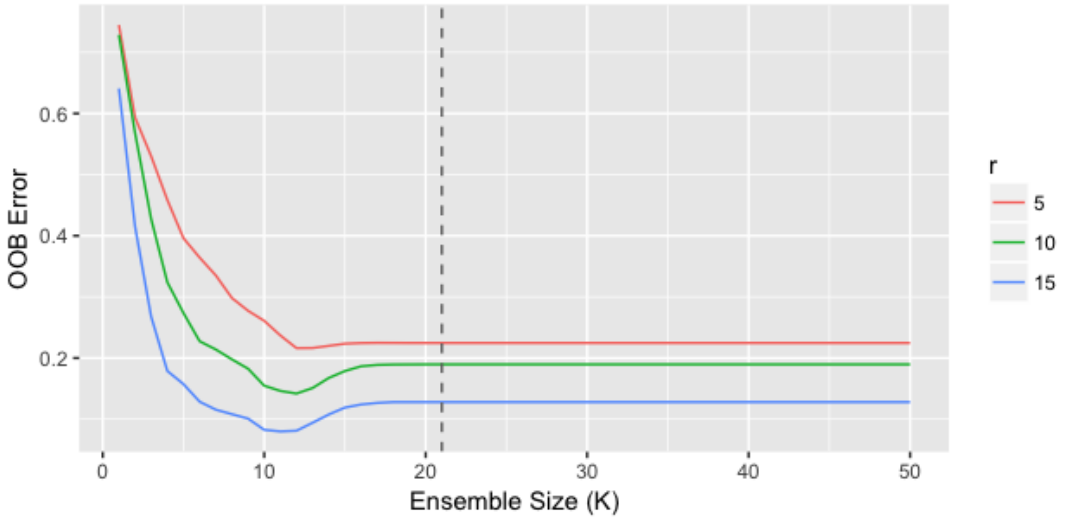

Figure 4.9: OOB error graph of the pen digits data, with $r = \{5, 10, 15\}$.

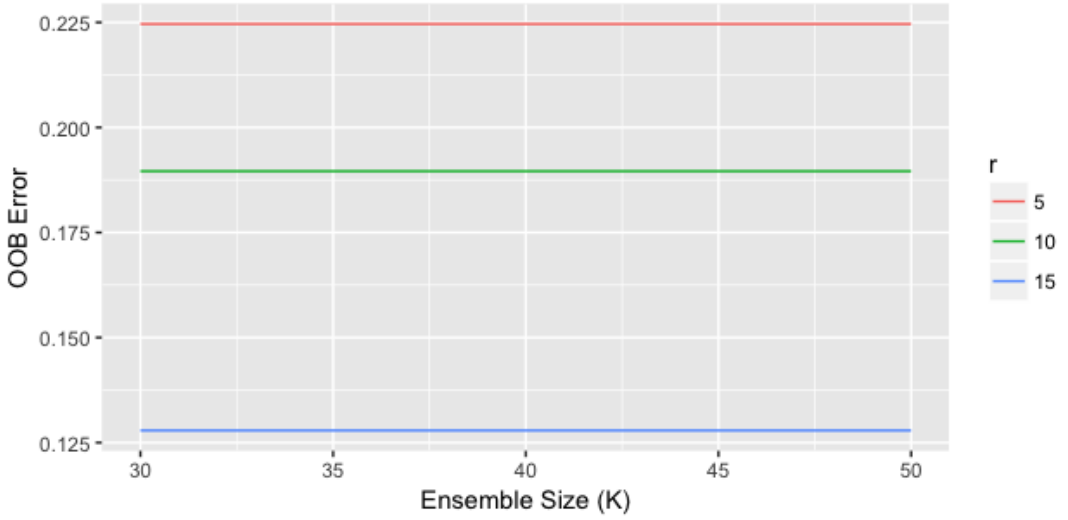

Figure 4.10: Tail of the OOB error graph of the pen digits data, with $r = \{5, 10, 15\}$.

$13, 14, 15, 16$. However, looking at Table 4.18, $r = 16$ appears to be the best, and $r = 14$ appears to be the second best, by large margins. The test errors from the 20% holdout set, are given in Table 4.19.

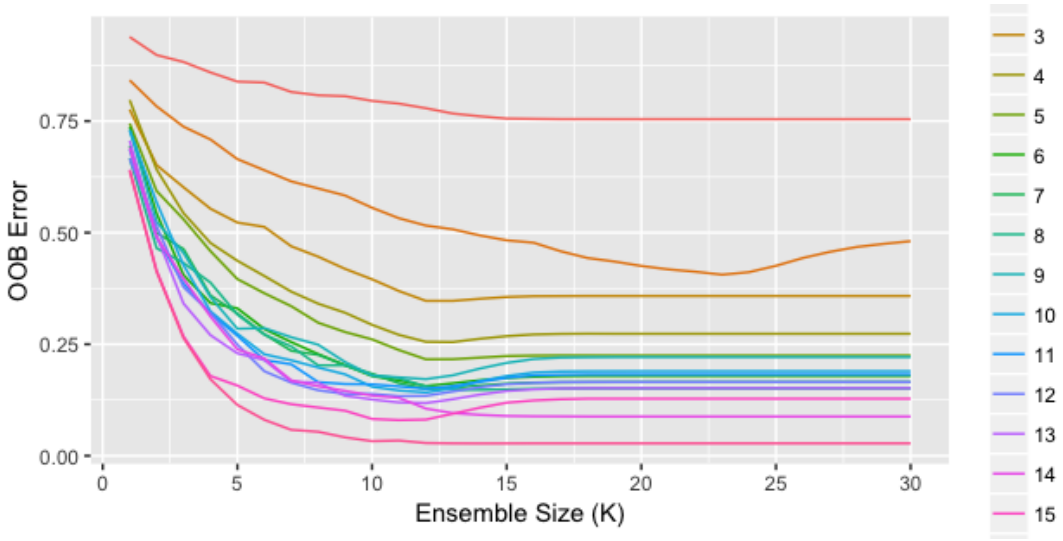

Figure 4.11: OOB error graph of the pen digits data, with $r = 1, \ldots, 16$.

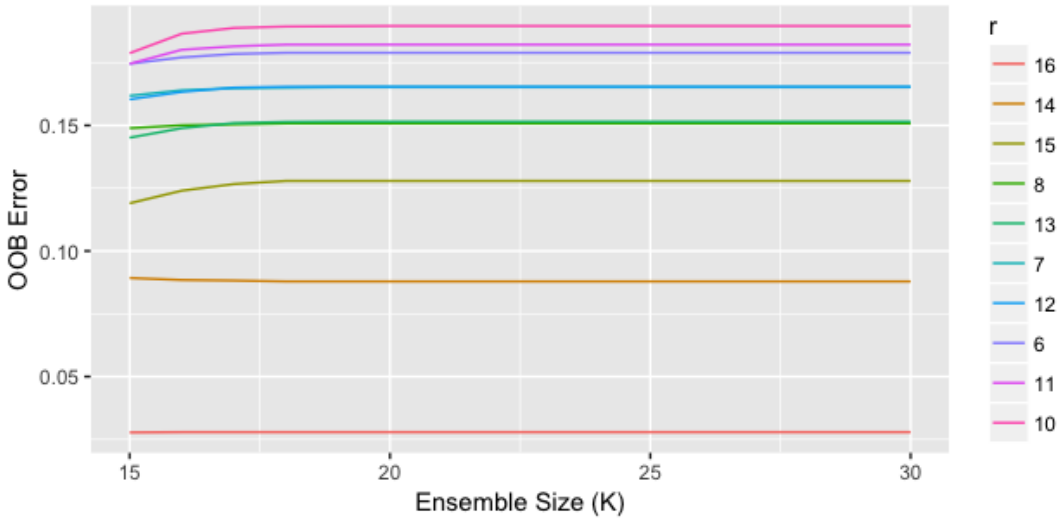

Figure 4.12: OOB error graph of the pen digits data: 10 lowest sequences.

Again, OOB error serves as an accurate predictor of which mixForests will perform the best when calculating test error using the holdout set. For this dataset it appears that the mixForest method does not score a lower test error than the default

Table 4.17: OOB errors for the pen digits data with $r = 1, \ldots, 16$ and $K = 30$.

# of Random Features	OOB Error
1	0.7543
2	0.4809
3	0.3581
4	0.2734
5	0.2246
6	0.1789
7	0.1652
8	0.1507
9	0.2211
10	0.1896
11	0.1822
12	0.1654
13	0.1515
14	0.0879
15	0.1279
16	0.0279

MDA method. A confusion matrix for the mixForest with $r = 16$ is shown in Table 4.20. Nonetheless, the mixForest method has advantages like variable importance scores, shown in Table 4.21.

Because the data were comprised of sequential coordinates, we can conclude from the variable importance scores in Table 4.21 that the first and last pen strokes indicate most clearly which digit was being written.

Finally, Table 4.22 shows the number of latent factors that each mixForest used.

Table 4.18: Sorted OOB errors for pen digits data with $r = 1, \ldots, 16$ and $K = 30$.

# of Random Features	OOB Error
16	0.0279
14	0.0879
15	0.1279
8	0.1507
13	0.1515
7	0.1652
12	0.1654
6	0.1789
11	0.1822
10	0.1896
9	0.2211
5	0.2246
4	0.2734
3	0.3581
2	0.4809
1	0.7543

Table 4.19: Test error comparison of models fitted to the pen-digits data.

Model	Test Error
MDA	0.0232
MixForest ($r = 14$)	0.0277
MixForest ($r = 16$)	0.0241

The best performing mixForest used models which used only $q = 10$. This represents a significant reduction in dimensionality.

Table 4.20: Pen digits data confusion matrix for mixForest with $r = 16$ and $K = 30$.

Predicted \ Actual	0	1	2	3	4	5	6	7	8	9
0	224	0	0	0	0	0	0	0	0	0
1	0	217	9	2	0	0	0	6	0	1
2	0	4	219	0	1	0	0	0	0	0
3	0	1	0	205	0	0	0	0	0	0
4	0	0	0	0	224	0	0	0	0	0
5	0	1	0	1	0	209	1	0	0	0
6	1	0	0	0	0	0	209	0	0	0
7	0	0	1	1	0	0	0	218	0	0
8	3	1	0	0	0	0	1	0	211	0
9	1	5	0	2	4	2	0	4	0	210

4.6 Discussion

The mixForest method appears to work quite well in comparison to MDA, in the context of using mixtures of factor analyzers to deal with high dimensional data. For the forest types mapping data mixForests yielded an increase of 19.05% classification accuracy compared to MDA. For the ionosphere data mixForests yielded an up to 2.86% increase in classification accuracy. However, for the pen-based handwritten digit data mixForests resulted in a decrease in classification accuracy of 0.45% when compared to MDA. Regardless of a nominal increase or decrease in accuracy, the mixForest method yields benefits above and beyond potential accuracy, including variable importance scores. As usual, the difficulty is in tuning the parameters.

For the mixForest method, there are only two parameters of concern: the number

Table 4.21: Pen digits data variable importance for mixForest with $r = 16$ and $K = 30$.

Variable	Raw Score	Z Score	P Value
15	759.47	13.86	0.0000
2	262.97	11.73	0.0000
4	513.70	11.32	0.0000
1	313.87	10.98	0.0000
16	779.33	9.77	0.0000
10	731.37	9.38	0.0000
14	910.23	7.11	0.0000
11	726.67	6.00	0.0000
12	683.70	5.93	0.0000
9	686.93	5.51	0.0000
6	847.53	5.50	0.0000
5	849.00	5.04	0.0000
7	666.90	5.03	0.0000
3	457.70	4.73	0.0000
13	301.53	4.36	0.0000
8	895.00	4.26	0.0000

Table 4.22: Number of latent factors (q) used by the mixForest with $r = 16$ for the pen-based handwritten digits data.

q	Frequency
10	30

of randomly selected features (r) and the ensemble size (K, to a lesser extent). The ensemble size has to be large enough for the OOB error to converge, signifying stability in the prediction results, but small enough to avoid large computation times.

In the data analysis and illustrations section, the evidence shows that OOB error converges after only 15 to 30 classifiers are added to a mixForest. The real issue is choosing r.

For different values of r, mixForests produce a wide range of OOB errors. The issue is that the pattern in the OOB error rates isn't predictable by the magnitude of r. Thus, fitting p mixForests is necessary to find the one which will perform the best. Even so, for a single set of data, it is possible that more than one 'sweet spot' exists, which makes the selection of r more difficult. The saving grace of the process is that the OOB errors accurately predicted the test errors. For the forest types mapping data with $p = 27$, a value of $r = 14$ performed the best. For the ionosphere data with $p = 34$, a value of $r = 25$ performed the best. For the pen-based handwritten digits recognition data with $p = 16$, a value of $r = 16$ performed the best. Empirically, it is difficult to maintain a default value of r, that might work in general practice.

Finally, the mixForest method was successful in reducing the dimension of the data. In the case of the forest types mapping data , the number of features is $p = 27$ and the most accurate mixForest ($r = 14$) was comprised mainly of classifiers using $q = 9$ or less latent factors. In the case of the ionosphere data, the number of features is $p = 34$ and the most accurate mixForest ($r = 25$) was comprised of classifiers with $q \leq 15$ latent variables, though 78% of the classifiers used $q = 2$ latent variables. In the case of the pen-based handwritten digit recognition data with $p = 16$, the most accurate mixForest ($r = 16$) was made only of classifiers using $q = 10$ latent variables.

Chapter 5

Conclusions

In this thesis, two types of mixForests were used to analyze three datasets each. The first is a mixForest based on the family of GPCMs and is applicable to lower dimensional data. The second is a mixForest based on the family of PGMMs and is applicable to the higher dimensional case. Let these be known as type I and type II mixForests, respectively. In both mixForest cases, computation time is significantly greater than for MDA. Arguably, the mixForests performed at least as well as MDA, with the caveat being the correct selection of the only real tuning parameter (r).

It was shown that the ensemble size K does not affect inference as long as K is chosen large enough that the OOB error sequence for the mixForest stabilizes. However, the selection of the number of randomly selected features, r, is critical to the performance of the mixForest. While OOB error proved to be a surprisingly good predictor of the final test errors, it still proved to be problematic to choose the value of r which always performed the best. Thus, it is possible to generate a

mixForest which will not perform as well as the optimal choice, though the difference is marginal.

The real justification for the extra computation time, in the cases of marginal performance increase or decrease, is the added inferential advantage that mixForests provide. Firstly, OOB error estimates provide an unbiased estimator for test error, which can be calculated during the generation of the mixForest. Secondly, variable importance is easily deduced from simple permutations of the data. Data reduction can be performed easily based on these scores for both types of mixForests, though the type II mixForest has an added advantage since dimensional reduction is a natural by product of factor analyzers. While it was not demonstrated herein, mixForests handle missing data easily since the missing values do not have to be imputed. Since each classifier in the mixForest is generated from a random selection of features, when a value is missing in the data, a consensus amongst the classifiers which do not use the missing feature can still be achieved. Performance would then depend on the importance of the missing feature. Future work will be needed to study this phenomenon, including the impact of missing data when calculating the OOB error estimate.

For the type II mixForest, the fitting procedure required numerous models to be fitted for each bootstrap replicate, based on a range of values for G and q, as well as all the model types in the PGMM family of models. This required a great deal of extra computation time, though in the final mixForest the chosen classifiers exhibited very little diversity in terms of model type and number of latent factors (q). While type II mixForests require far fewer classifiers before the OOB error sequence

stabilizes, computation time can still be saved with a precursory analysis of the latent structure so that a smaller range of q is tried.

There are a number of possibilities for future work. This includes the development of new types of mixForests. The FMM literature is rich with mixtures based on many different probabilistic structures. Each of these can lead to a new type of mixForest. In addition to this, a heterogeneous mixForest is also possible, where the classifiers can be fitted using a variety of base learning algorithms. In the heterogenous mixForest, a number of different model types would be fitted for each bootstrap replicate and the best one would be chosen by a model selection criterion such as the BIC.

Another possibility for future work is the development of an algorithm that can self-select the number of randomly selected features (r). Classifiers can be added to this type of self-selecting mixForest based on an overall decrease in the OOB error. In this way, an optimal value for r can be chosen. The end result would be a mixForest using classifiers with a variety of values for r, with the most critical value(s) of r having a greater frequency in the mixForest.

As a final note, there is one data analysis that appears to be quite significant. With the exception of one single dataset, MDA produced test errors which can be considered to be reasonably small. However, for the forest types mapping data MDA produced a test error of 27.62%. When mixForests were applied to this data, the test error decreased to 8.57%. In fact, there was a wide range of values for r which produced mixForests which also provided significant decreases in test error when compared to MDA. It is a definite possibility that in cases when MDA does not

perform well, mixForests will provide significant improvements.

Appendix A

Tables for Ionosphere Data

Tables for the ionosphere data are provided here as Tables A.1 to A.6.

Table A.1: OOB errors for the ionosphere data with $r = 1, \ldots, 17$ and $K = 50$.

# of Random Features	OOB Error
1	0.3679
2	0.2903
3	0.2643
4	0.2366
5	0.1835
6	0.1577
7	0.1661
8	0.1183
9	0.1223
10	0.1547
11	0.1151
12	0.1111
13	0.1434
14	0.1505
15	0.1685
16	0.1547
17	0.1685

Table A.2: OOB errors for the ionosphere data with $r = 18, \ldots, 33$ and $K = 50$.

# of Random Features	OOB Error
18	0.1685
19	0.1475
20	0.1750
21	0.1788
22	0.1398
23	0.1505
24	0.1864
25	0.1286
26	0.1357
27	0.1470
28	0.1720
29	0.1475
30	0.1685
31	0.1393
32	0.1821
33	0.1536

Table A.3: Part 1: sorted OOB errors for the ionosphere data.

# of Random Features	OOB Error
12	0.1111
11	0.1151
8	0.1183
9	0.1223
25	0.1286
26	0.1357
31	0.1393
22	0.1398
13	0.1434
27	0.1470
19	0.1475
29	0.1475
14	0.1505
23	0.1505
33	0.1536
10	0.1547
16	0.1547

Table A.4: Part 2: sorted OOB errors for the ionosphere data.

# of Random Features	OOB Error
6	0.1577
7	0.1661
15	0.1685
17	0.1685
18	0.1685
30	0.1685
28	0.1720
20	0.1750
21	0.1788
32	0.1821
5	0.1835
24	0.1864
4	0.2366
3	0.2643
2	0.2903
1	0.3679

Table A.5: Part 1: Ionosphere data variable importance for mixForest with $r = 25$ and $K = 50$.

Variable	Raw Score	Z Score	P Value
2	5.81	3.06	0.0022
1	5.88	2.42	0.0156
8	5.75	1.94	0.0519
4	4.25	1.78	0.0746
7	6.98	1.75	0.0804
10	4.52	1.72	0.0860
16	1.55	1.39	0.1652
6	3.29	1.22	0.2220
20	1.86	1.11	0.2679
32	1.31	0.83	0.4062
13	3.10	0.77	0.4398
24	1.85	0.68	0.4944
30	1.51	0.64	0.5245
12	1.02	0.62	0.5332
11	1.88	0.61	0.5430
9	1.98	0.58	0.5625
26	3.00	0.57	0.5677

Table A.6: Part 2: Ionosphere data variable importance for mixForest with $r = 25$ and $K = 50$.

Variable	Raw Score	Z Score	P Value
22	0.66	0.47	0.6383
28	5.17	0.44	0.6589
27	0.79	0.42	0.6709
25	0.33	0.38	0.7003
3	0.90	0.38	0.7068
21	0.28	0.31	0.7590
18	0.27	0.30	0.7617
5	0.43	0.29	0.7709
33	0.20	0.28	0.7758
17	0.18	0.23	0.8162
23	0.18	0.23	0.8202
19	0.13	0.22	0.8244
14	0.12	0.20	0.8454
15	0.10	0.13	0.8935
29	0.17	0.10	0.9171
31	-0.09	-0.17	0.8628

Bibliography

Alimoglu, F., Alpaydin, E., and Denizhan, Y. (1996). Combining multiple classifiers for pen-based handwritten digit recognition.

Andrews, J. L. and McNicholas, P. D. (2011). Mixtures of modified t-factor analyzers for model-based clustering, classification, and discriminant analysis. *Journal of Statistical Planning and Inference*, **141**(4), 1479–1486.

Archer, K. J. and Kimes, R. V. (2008). Empirical characterization of random forest variable importance measures. *Computational Statistics & Data Analysis*, **52**(4), 2249–2260.

Banfield, J. D. and Raftery, A. E. (1993). Model-based Gaussian and non-Gaussian clustering. *Biometrics*, **49**(3), 803–821.

Biau, G. (2012). Analysis of a random forests model. *Journal of Machine Learning Research*, **13**(Apr), 1063–1095.

Breiman, L. (1996a). Bagging predictors. *Machine learning*, **24**(2), 123–140.

Breiman, L. (1996b). Out-of-bag estimation. Technical report, Statistics Department, University of California Berkeley, Berkeley CA 94708.

Breiman, L. (2001). Random forests. *Machine learning*, **45**(1), 5–32.

Breiman, L. and Cutler, A. (2004). Random forests (2011). *URL http://www. stat. berkeley. edu/˜ breiman/RandomForests/cc_ home. htm, Version*, **5**.

Breiman, L. *et al.* (1996). Heuristics of instability and stabilization in model selection. *The Annals of Statistics*, **24**(6), 2350–2383.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and Regression Trees*. CRC press, Boca Raton, FL.

Browne, R. P. and McNicholas, P. D. (2015). A mixture of generalized hyperbolic distributions. *Canadian Journal of Statistics*, **43**(2), 176–198.

Celeux, G. and Govaert, G. (1995). Gaussian parsimonious clustering models. *Pattern recognition*, **28**(5), 781–793.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, **39**(1), 1–22.

Fern, X. Z. (2012). Unsupervised learning: Model selection and evaluation. *http://web.engr.oregonstate.edu/ xfern/classes/cs534/notes/Unsupervised-model-11.pdf*.

Forina, M., Armanino, C., Castino, M., and Ubigli, M. (1986). Multivariate data analysis as a discriminating method of the origin of wines. *Vitis*, **25**(3), 189–201.

Fraley, C. and Raftery, A. E. (2002). Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, **97**(458), 611–631.

Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The Elements of Statistical Learning*, volume 1. Springer, New York.

Hssina, B., Merbouha, A., Ezzikouri, H., and Erritali, M. (2014). A comparative study of decision tree ID3 and C4. 5. *International Journal of Advanced Computer Science and Applications*, **4**(2), 0–0.

Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of Classification*, **2**(1), 193–218.

Hunt, E. B. (1962). *Concept Formation: An Information Processing Problem*. John Wiley & Sons, Inc., New York.

Jeffreys, H. (1961). *Theory of Probability*. Oxford University Press, Oxford, UK.

Johnson, B., Tateishi, R., and Xie, Z. (2012). Using geographically weighted variables for image classification. *Remote Sensing Letters*, **3**(6), 491–499.

Kass, G. V. (1980). An exploratory technique for investigating large quantities of categorical data. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, **29**(2), 119–127.

Lawley, D. N. and Maxwell, A. E. (1962). Factor analysis as a statistical method. *Journal of the Royal Statistical Society. Series D (The Statistician)*, **12**(3), 209–229.

Lichman, M. (2013). UCI machine learning repository. https://archive.ics.uci.edu/.

McLachlan, G. and Peel, D. (2000a). *Finite Mixture Models*. Wiley, New York.

McLachlan, G. and Peel, D. (2000b). Mixtures of factor analyzers. In *Proceedings of the Seventeenth International Conference on Machine Learning*. Citeseer.

McLachlan, G. J., Bean, R., and Jones, L. B.-T. (2007). Extension of the mixture of factor analyzers model to incorporate the multivariate t-distribution. *Computational Statistics & Data Analysis*, **51**(11), 5327–5338.

McNicholas, P. D. (2010). Model-based classification using latent Gaussian mixture models. *Journal of Statistical Planning and Inference*, **140**(5), 1175–1181.

McNicholas, P. D. (2016). *Mixture Model-Based Classification*. CRC Press, Boca Raton, FL.

McNicholas, P. D. and Murphy, T. B. (2008). Parsimonious Gaussian mixture models. *Statistics and Computing*, **18**(3), 285–296.

McNicholas, P. D. and Murphy, T. B. (2010). Model-based clustering of microarray expression data via latent Gaussian mixture models. *Bioinformatics*, **26**(21), 2705–2712.

McNicholas, P. D., Murphy, T. B., McDaid, A. F., and Frost, D. (2010). Serial and parallel implementations of model-based clustering via parsimonious Gaussian mixture models. *Computational Statistics & Data Analysis*, **54**(3), 711–723.

McNicholas, P. D., ElSherbiny, A., McDaid, A. F., and Murphy, T. B. (2018). *pgmm: Parsimonious Gaussian Mixture Models*. R package version 1.2.3.

McNicholas, S. M., McNicholas, P. D., and Browne, R. P. (2017). A mixture of variance-gamma factor analyzers. In *Big and Complex Data Analysis*, pages 369–385. Springer.

Meng, X.-L. and Rubin, D. B. (1993). Maximum likelihood estimation via the ECM algorithm: A general framework. *Biometrika*, **80**(2), 267–278.

Meng, X.-L. and Van Dyk, D. (1997). The EM algorithm – an old folk-song sung to a fast new tune. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **59**(3), 511–567.

Milborrow, S. and Milborrow, M. S. (2018). *rpart.plot: Plot 'rpart' Models: An Enhanced Version of 'plot.rpart'*. R package version 3.0.6.

Morgan, J. N. and Messenger, R. C. (1973). THAID: A sequential analysis program for the analysis of nominal scale dependent variables. *Institute for Social Research, University of Michigan*.

Morgan, J. N. and Sonquist, J. A. (1963). Problems in the analysis of survey data, and a proposal. *Journal of the American Statistical Association*, **58**(302), 415–434.

Murray, P. M., Browne, R. P., and McNicholas, P. D. (2014). Mixtures of skew-t factor analyzers. *Computational Statistics & Data Analysis*, **77**, 326–335.

Murray, P. M., Browne, R. P., and McNicholas, P. D. (2017). Hidden truncation hyperbolic distributions, finite mixtures thereof, and their application for clustering. *Journal of Multivariate Analysis*, **161**, 141–156.

Newcomb, S. (1886). A generalized theory of the combination of observations so as to obtain the best result. *American Journal of Mathematics*, pages 343–366.

Pearson, K. (1894). Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society of London. A*, **185**, 71–110.

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, **1**(1), 81–106.

Quinlan, J. R. (1993). C4.5: Programs for machine learning. *Machine Learning*, **16**(3), 235–240.

Quinlan, J. R. (1997). C5.0 data mining tool. https://rulequest.com/see5-info.html.

Raftery, A. E. (1995). Bayesian model selection in social research. *Sociological Methodology*, **25**, 111–164.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, **66**(336), 846–850.

Rohra, J. G., Perumal, B., Narayanan, S. J., Thakur, P., and Bhatt, R. B. (2017). User localization in an indoor environment using fuzzy hybrid of particle swarm optimization & gravitational search algorithm with neural networks. In *Proceedings*

of Sixth International Conference on Soft Computing for Problem Solving, pages 286–295. Springer.

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, **6**(2), 461–464.

Scrucca, L., Fop, M., Murphy, T. B., and Raftery, A. E. (2016). mclust 5: Clustering, classification and density estimation using Gaussian finite mixture models. *The R Journal*, **8**(1), 289.

Sigillito, V. G., Wing, S. P., Hutton, L. V., and Baker, K. B. (1989). Classification of radar returns from the ionosphere using neural networks. *Johns Hopkins APL Technical Digest*, **10**(3), 262–266.

Silva, A., Rothstein, S. J., McNicholas, P. D., and Subedi, S. (2017). A multivariate Poisson-log normal mixture model for clustering transcriptome sequencing data. *arXiv preprint arXiv:1711.11190*.

Smith, J. W., Everhart, J., Dickson, W., Knowler, W., and Johannes, R. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the Annual Symposium on Computer Application in Medical Care*, page 261. American Medical Informatics Association.

Therneau, T., Atkinson, B., and Ripley, B. (2018). *rpart: Recursive Partitioning and Regression Trees*. R package version 4.1-13.

Titterington, D., Smith, A., and Makov, U. (1985). *Statistical Analysis of Finite Mixture Models*. Wiley, Chichester, UK.

Webb, A. R. (2000). Gamma mixture models for target recognition. *Pattern Recognition*, **33**(12), 2045–2054.

Wei, Y. and McNicholas, P. D. (2015). Mixture model averaging for clustering. *Advances in Data Analysis and Classification*, **9**(2), 197–217.

Wikipedia (2019). Advanced spaceborne thermal emission and reflection radiometer.

Woodbury, M. A. (1950). Inverting modified matrices. Statistical Research Group, Memorandum Report 42, Princeton University, Princeton, NJ.

Zhou, H. and Lange, K. L. (2010). On the bumpy road to the dominant mode. *Scandinavian Journal of Statistics*, **37**(4), 612–631.